

# TMS320F28004x Microcontrollers

## Technical Reference Manual



Literature Number: SPRUI33C  
November 2015–Revised August 2019

<b>Preface</b> .....	<b>80</b>
<b>1 C2000 Software Support</b> .....	<b>81</b>
1.1 Introduction.....	82
1.2 C2000Ware Structure.....	82
1.3 Documentation.....	82
1.4 Devices .....	82
1.5 Libraries .....	82
1.6 Code Composer Studio.....	82
1.7 PinMUX Tool.....	83
<b>2 C28x Processor</b> .....	<b>84</b>
2.1 Introduction.....	85
2.2 Features .....	85
2.3 Floating-Point Unit .....	85
2.4 Trigonometric Math Unit .....	85
2.5 Viterbi, Complex Math and CRC Unit (VCU) .....	86
<b>3 TMS320F28004x System Control</b> .....	<b>87</b>
3.1 Introduction.....	88
3.2 Power Management.....	88
3.2.1 Internal 1.2-V Switching Regulator (DC-DC) .....	88
3.3 Device Identification and Configuration Registers .....	89
3.4 Resets .....	89
3.4.1 Reset Sources .....	89
3.4.2 External Reset (XRS).....	90
3.4.3 Power-On Reset (POR) .....	90
3.4.4 Debugger Reset (SYSRS) .....	90
3.4.5 Watchdog Reset (WDRS).....	90
3.4.6 NMI Watchdog Reset (NMIWDRS) .....	90
3.4.7 DCSM Safe Code Copy Reset (SCCRESET).....	90
3.5 Peripheral Interrupts .....	91
3.5.1 Interrupt Concepts.....	91
3.5.2 Interrupt Architecture.....	91
3.5.3 Interrupt Entry Sequence.....	92
3.5.4 Configuring and Using Interrupts.....	93
3.5.5 PIE Channel Mapping .....	95
3.5.6 Vector Tables .....	96
3.6 Exceptions and Non-Maskable Interrupts .....	102
3.6.1 Configuring and Using NMIs .....	102
3.6.2 Emulation Considerations .....	102
3.6.3 NMI Sources.....	102
3.6.4 Illegal Instruction Trap (ITRAP) .....	103
3.6.5 Error Pin.....	103
3.7 Clocking .....	104
3.7.1 Clock Sources.....	104
3.7.2 Derived Clocks .....	107
3.7.3 Device Clock Domains .....	107

3.7.4	XCLKOUT.....	108
3.7.5	Clock Connectivity .....	108
3.7.6	Clock Source and PLL Setup .....	110
3.7.7	Using an External Crystal or Resonator .....	110
3.7.8	Using an External Oscillator.....	111
3.7.9	Choosing PLL Settings.....	111
3.7.10	System Clock Setup.....	111
3.7.11	Clock Configuration Examples .....	112
3.7.12	Missing Clock Detection .....	112
3.8	32-Bit CPU Timers 0/1/2.....	114
3.9	Watchdog Timer .....	115
3.9.1	Servicing the Watchdog Timer .....	115
3.9.2	Minimum Window Check .....	116
3.9.3	Watchdog Reset or Watchdog Interrupt Mode.....	116
3.9.4	Watchdog Operation in Low Power Modes .....	117
3.9.5	Emulation Considerations .....	117
3.10	Low Power Modes .....	118
3.10.1	IDLE .....	118
3.10.2	Guidelines on Software Emulation of STANDBY Mode .....	118
3.10.3	HALT .....	119
3.10.4	Flash Power-down Considerations .....	119
3.11	Memory Controller Module .....	121
3.11.1	Functional Description .....	121
3.12	Flash and OTP Memory .....	128
3.12.1	Features.....	128
3.12.2	Flash Tools .....	128
3.12.3	Default Flash Configuration .....	129
3.12.4	Flash Bank, OTP and Pump .....	129
3.12.5	Flash Module Controller (FMC).....	129
3.12.6	Flash and OTP and Wakeup Power-Down Modes.....	130
3.12.7	Flash and OTP Performance.....	131
3.12.8	Flash Access Interface.....	131
3.12.9	Erase/Program Flash.....	134
3.12.10	Error Correction Code (ECC) Protection.....	135
3.12.11	Reserved Locations Within Flash and OTP .....	138
3.12.12	Procedure to Change the Flash Control Registers .....	138
3.13	Dual Code Security Module (DCSM).....	139
3.13.1	Functional Description .....	139
3.13.2	C Code Example to get Zone Select Block Addr for Zone1 in BANK0.....	147
3.13.3	Flash and OTP Erase/Program .....	147
3.13.4	Safe Copy Code .....	147
3.13.5	SafeCRC.....	148
3.13.6	CSM Impact on Other On-Chip Resources .....	148
3.13.7	Incorporating Code Security in User Applications.....	150
3.14	F28004x System Control Registers .....	155
3.14.1	System Control Base Addresses .....	155
3.14.2	ACCESS_PROTECTION_REGS Registers.....	156
3.14.3	CLK_CFG_REGS Registers .....	179
3.14.4	CPU_SYS_REGS Registers .....	195
3.14.5	CPUTIMER_REGS Registers.....	236
3.14.6	DCSM_BANK0_Z1_REGS Registers.....	243
3.14.7	DCSM_BANK0_Z2_REGS Registers.....	268
3.14.8	DCSM_BANK1_Z1_REGS Registers.....	289

3.14.9	DCSM_BANK1_Z2_REGS Registers .....	300
3.14.10	DCSM_COMMON_REGS Registers .....	311
3.14.11	DEV_CFG_REGS Registers.....	324
3.14.12	DMA_CLA_SRC_SEL_REGS Registers.....	352
3.14.13	FLASH_CTRL_REGS Registers .....	359
3.14.14	FLASH_ECC_REGS Registers .....	369
3.14.15	MEM_CFG_REGS Registers .....	392
3.14.16	MEMORY_ERROR_REGS Registers .....	438
3.14.17	NMI_INTERRUPT_REGS Registers .....	455
3.14.18	PERIPH_AC_REGS Registers .....	464
3.14.19	PIE_CTRL_REGS Registers .....	518
3.14.20	ROM_PREFETCH_REGS Registers .....	570
3.14.21	ROM_WAIT_STATE_REGS Registers.....	571
3.14.22	UID_REGS Registers.....	572
3.14.23	WD_REGS Registers .....	581
3.14.24	XINT_REGS Registers .....	588
3.14.25	Register to Driverlib Function Mapping.....	597
<b>4</b>	<b>ROM Code and Peripheral Booting .....</b>	<b>608</b>
4.1	Introduction .....	609
4.2	Device Boot Philosophy.....	609
4.3	Device Boot Modes.....	609
4.3.1	Configuring Alternate Boot Mode Pins .....	610
4.3.2	Configuring Alternate Boot Mode Options.....	611
4.3.3	Boot Mode Example Use Cases.....	612
4.4	Device Boot Flow Diagrams .....	614
4.4.1	Emulation Boot Flow Diagram .....	615
4.4.2	Standalone Boot Flow Diagram.....	616
4.5	Device Reset and Exception Handling.....	617
4.5.1	Reset Causes and Handling.....	617
4.5.2	Exceptions and Interrupts Handling .....	617
4.6	Boot ROM Description .....	618
4.6.1	Boot ROM Registers .....	618
4.6.2	Boot ROM User OTP .....	618
4.6.3	Entry Points.....	618
4.6.4	Wait Points.....	619
4.6.5	Memory Maps .....	619
4.6.6	ROM Tables .....	621
4.6.7	Boot Modes.....	622
4.6.8	Boot Data Stream Structure .....	634
4.6.9	GPIO Assignments .....	636
4.6.10	DCSM Usage.....	637
4.6.11	Clock Initialization.....	638
4.6.12	Boot Status Information .....	638
4.6.13	ROM Version .....	639
4.7	Application Notes for Using the Bootloaders.....	640
4.7.1	The C2000 Hex Utility .....	640
<b>5</b>	<b>Control Law Accelerator (CLA) .....</b>	<b>642</b>
5.1	Introduction .....	643
5.2	Features.....	643
5.3	CLA Interface.....	645
5.3.1	CLA Memory .....	645
5.3.2	CLA Memory Bus .....	646
5.3.3	Shared Peripherals and EALLOW Protection .....	646



5.3.4	CLA Tasks and Interrupt Vectors .....	647
5.3.5	CLA Software Interrupt to CPU .....	650
5.4	CLA, DMA, and CPU Arbitration .....	650
5.4.1	CLA Message RAM .....	650
5.5	CLA Configuration and Debug .....	652
5.5.1	Building a CLA Application .....	652
5.5.2	Typical CLA Initialization Sequence .....	652
5.5.3	Debugging CLA Code .....	653
5.5.4	CLA Illegal Opcode Behavior .....	655
5.5.5	Resetting the CLA .....	655
5.6	Pipeline.....	657
5.6.1	Pipeline Overview.....	657
5.6.2	CLA Pipeline Alignment.....	657
5.6.3	Parallel Instructions.....	662
5.7	Instruction Set .....	663
5.7.1	Instruction Descriptions .....	663
5.7.2	Addressing Modes and Encoding.....	665
5.7.3	Instructions .....	667
5.8	CLA Registers.....	781
5.8.1	CLA Base Addresses.....	781
5.8.2	CLA_ONLY_REGS Registers .....	782
5.8.3	CLA_SOFTINT_REGS Registers.....	791
5.8.4	CLA_REGS Registers.....	795
<b>6</b>	<b>Dual-Clock Comparator (DCC).....</b>	<b>846</b>
6.1	Introduction .....	847
6.2	Features.....	847
6.2.1	Block Diagram.....	847
6.3	Module Operation .....	847
6.3.1	Configuring DCC Counters .....	847
6.3.2	Single-Shot Measurement Mode .....	848
6.4	Interrupts .....	849
6.5	DCC Registers .....	850
6.5.1	DCC Base Addresses .....	850
6.5.2	DCC_REGS Registers .....	851
6.5.3	Register to Driverlib Function Mapping .....	863
<b>7</b>	<b>CLA Program ROM CRC (CLAPROMCRC).....</b>	<b>864</b>
7.1	Overview .....	865
7.2	Functional Description.....	865
7.2.1	Start Address .....	865
7.2.2	Seed .....	865
7.2.3	Halt .....	866
7.2.4	Result and Comparison .....	866
7.3	CLAPROM Registers.....	867
7.3.1	CLAPROM Base Addresses .....	867
7.3.2	CLA_PROM_CRC32_REGS Registers.....	868
<b>8</b>	<b>General-Purpose Input/Output (GPIO) .....</b>	<b>880</b>
8.1	Introduction .....	881
8.2	Configuration Overview .....	882
8.3	Digital General-Purpose I/O Control.....	883
8.4	Input Qualification.....	884
8.4.1	No Synchronization (Asynchronous Input) .....	884
8.4.2	Synchronization to SYSCLKOUT Only.....	884
8.4.3	Qualification Using a Sampling Window .....	884

8.5	GPIO and Peripheral Muxing.....	887
8.6	Internal Pullup Configuration Requirements.....	893
8.7	GPIO Registers .....	894
8.7.1	GPIO Base Addresses .....	894
8.7.2	GPIO_CTRL_REGS Registers.....	895
8.7.3	GPIO_DATA_REGS Registers .....	967
8.7.4	Register to Driverlib Function Mapping .....	986
<b>9</b>	<b>Crossbar (X-BAR) .....</b>	<b>989</b>
9.1	Introduction .....	990
9.2	ePWM, CLB, and GPIO Output X-BAR .....	991
9.2.1	ePWM X-BAR .....	991
9.2.2	CLB X-BAR .....	993
9.2.3	GPIO Output X-BAR.....	996
9.2.4	X-BAR Flags.....	997
9.3	XBAR Registers.....	999
9.3.1	XBAR Base Addresses .....	999
9.3.2	XBAR_REGS Registers .....	1000
9.3.3	INPUT_XBAR_REGS Registers .....	1036
9.3.4	OUTPUT_XBAR_REGS Registers .....	1055
9.3.5	EPWM_XBAR_REGS Registers .....	1156
9.3.6	CLB_XBAR_REGS Registers .....	1249
9.3.7	Register to Driverlib Function Mapping.....	1358
<b>10</b>	<b>Direct Memory Access (DMA) .....</b>	<b>1363</b>
10.1	Introduction.....	1364
10.2	Features .....	1364
10.3	Architecture .....	1365
10.3.1	Block Diagram .....	1365
10.3.2	Peripheral Interrupt Event Trigger Sources.....	1365
10.3.3	DMA Bus .....	1369
10.4	Address Pointer and Transfer Control.....	1369
10.5	Pipeline Timing and Throughput .....	1374
10.6	CPU and CLA Arbitration.....	1375
10.7	Channel Priority.....	1376
10.7.1	Round-Robin Mode .....	1376
10.7.2	Channel 1 High Priority Mode .....	1377
10.8	Overrun Detection Feature.....	1377
10.9	DMA Registers.....	1378
10.9.1	DMA Base Addresses .....	1378
10.9.2	DMA_REGS Registers .....	1379
10.9.3	DMA_CH_REGS Registers.....	1384
10.9.4	Register to Driverlib Function Mapping.....	1412
<b>11</b>	<b>Embedded Real-time Analysis and Diagnostic (ERAD) .....</b>	<b>1414</b>
11.1	Introduction.....	1415
11.2	Enhanced Bus Comparator Unit.....	1415
11.2.1	Enhanced Bus Comparator Unit Operations .....	1416
11.3	System Event Counter Unit .....	1416
11.3.1	System Event Counter Modes .....	1417
11.3.2	Reset on Event .....	1419
11.3.3	Operation Conditions .....	1419
11.4	ERAD Ownership, Initialization and Reset.....	1419
11.5	ERAD Programming Sequence .....	1419
11.5.1	Hardware Breakpoint and Hardware Watch Point Programming Sequence .....	1420

11.5.2	Timer and Counter Programming Sequence .....	1420
11.6	ERAD Registers .....	1422
11.6.1	ERAD Base Addresses.....	1422
11.6.2	ERAD_GLOBAL_REGS Registers.....	1423
11.6.3	ERAD_HWBP_REGS Registers .....	1432
11.6.4	ERAD_COUNTER_REGS Registers .....	1439
<b>12</b>	<b>Analog Subsystem .....</b>	<b>1448</b>
12.1	Introduction.....	1449
12.1.1	Features .....	1449
12.1.2	Block Diagram .....	1449
12.1.3	Lock Registers.....	1457
12.2	Analog Subsystem Registers .....	1458
12.2.1	Analog Subsystem Base Addresses.....	1458
12.2.2	ANALOG_SUBSYS_REGS Registers.....	1459
<b>13</b>	<b>Analog-to-Digital Converter (ADC).....</b>	<b>1476</b>
13.1	Introduction.....	1477
13.2	ADC Features.....	1477
13.3	ADC Block Diagram.....	1478
13.4	ADC Configurability .....	1478
13.4.1	Clock Configuration .....	1479
13.4.2	Resolution .....	1479
13.4.3	Voltage Reference .....	1479
13.4.4	Signal Mode.....	1480
13.4.5	Expected Conversion Results .....	1480
13.4.6	Interpreting Conversion Results .....	1480
13.5	SOC Principle of Operation .....	1481
13.5.1	SOC Configuration .....	1481
13.5.2	Trigger Operation .....	1481
13.5.3	ADC Acquisition (Sample and Hold) Window .....	1482
13.5.4	ADC Input Models.....	1482
13.5.5	Channel Selection.....	1483
13.6	SOC Configuration Examples .....	1484
13.6.1	Single Conversion from ePWM Trigger .....	1484
13.6.2	Oversampled Conversion from ePWM Trigger.....	1484
13.6.3	Multiple Conversions from CPU Timer Trigger .....	1485
13.6.4	Software Triggering of SOCs .....	1486
13.7	ADC Conversion Priority .....	1486
13.8	Burst Mode .....	1489
13.8.1	Burst Mode Example.....	1489
13.8.2	Burst Mode Priority Example .....	1489
13.9	EOC and Interrupt Operation .....	1491
13.9.1	Interrupt Overflow .....	1491
13.9.2	Continue to Interrupt Mode .....	1492
13.9.3	Early Interrupt Configuration Mode .....	1492
13.10	Post-Processing Blocks .....	1492
13.10.1	PPB Offset Correction.....	1493
13.10.2	PPB Error Calculation .....	1494
13.10.3	PPB Limit Detection and Zero-Crossing Detection.....	1494
13.10.4	PPB Sample Delay Capture .....	1495
13.11	Opens/Shorts Detection Circuit (OSDETECT) .....	1496
13.11.1	Implementation.....	1497
13.11.2	Detecting an Open Input Pin .....	1497
13.11.3	Detecting a Shorted Input Pin.....	1497

13.12	Power-Up Sequence .....	1497
13.13	ADC Calibration .....	1498
13.13.1	ADC Zero Offset Calibration .....	1498
13.14	ADC Timings .....	1498
13.14.1	ADC Timing Diagrams .....	1499
13.15	Additional Information .....	1501
13.15.1	Ensuring Synchronous Operation .....	1501
13.15.2	Choosing an Acquisition Window Duration .....	1505
13.15.3	Achieving Simultaneous Sampling .....	1506
13.15.4	Designing an External Reference Circuit .....	1506
13.15.5	Internal Temperature Sensor .....	1508
13.16	ADC Registers .....	1509
13.16.1	ADC Base Addresses .....	1509
13.16.2	ADC_REGS Registers .....	1510
13.16.3	ADC_RESULT_REGS Registers .....	1631
13.16.4	Register to Driverlib Function Mapping .....	1652
<b>14</b>	<b>Programmable Gain Amplifier (PGA) .....</b>	<b>1656</b>
14.1	Programmable Gain Amplifier (PGA) Overview .....	1657
14.1.1	Features .....	1657
14.1.2	Block Diagram .....	1657
14.2	Linear Output Range .....	1658
14.3	Gain Modes .....	1658
14.4	External Filtering .....	1658
14.5	Error Calibration .....	1659
14.5.1	Offset Error .....	1659
14.5.2	Gain Error .....	1659
14.6	Ground Routing .....	1660
14.7	Enabling and Disabling the PGA Clock .....	1661
14.8	Lock Register .....	1661
14.9	Examples .....	1662
14.9.1	Direct Amplifier .....	1662
14.9.2	RC Filter .....	1662
14.10	Analog Front End Integration .....	1662
14.10.1	ADC .....	1663
14.10.2	CMPSS .....	1663
14.10.3	Buffered DAC .....	1663
14.10.4	Alternate Functions .....	1664
14.11	PGA Registers .....	1665
14.11.1	PGA Base Addresses .....	1665
14.11.2	PGA_REGS Registers .....	1666
<b>15</b>	<b>Buffered Digital-to-Analog Converter (DAC) .....</b>	<b>1674</b>
15.1	Introduction .....	1675
15.1.1	Features .....	1675
15.1.2	Block Diagram .....	1675
15.2	Using the DAC .....	1675
15.2.1	Initialization Sequence .....	1676
15.2.2	DAC Offset Adjustment .....	1676
15.2.3	EPWMSYNCPER Signal .....	1677
15.3	Lock Registers .....	1677
15.4	DAC Registers .....	1678
15.4.1	DAC Base Addresses .....	1678
15.4.2	DAC_REGS Registers .....	1679
15.4.3	Register to Driverlib Function Mapping .....	1687

<b>16</b>	<b>Comparator Subsystem (CMPSS)</b> .....	<b>1688</b>
16.1	Introduction.....	1689
16.2	Features .....	1689
16.2.1	Block Diagram .....	1689
16.3	Comparator.....	1690
16.4	Reference DAC .....	1690
16.5	Ramp Generator.....	1692
16.5.1	Ramp Generator Overview .....	1692
16.5.2	Ramp Generator Behavior .....	1692
16.5.3	Ramp Generator Behavior at Corner Cases.....	1693
16.6	Digital Filter .....	1694
16.6.1	Filter Initialization Sequence .....	1695
16.7	Using the CMPSS .....	1695
16.7.1	LATCHCLR, EPWMSYNCPER and EPWMBLANK Signals .....	1695
16.7.2	Synchronizer, Digital Filter and Latch Delays .....	1695
16.7.3	Calibrating the CMPSS .....	1696
16.7.4	Enabling and Disabling the CMPSS Clock .....	1696
16.8	CMPSS Registers .....	1697
16.8.1	CMPSS Base Addresses.....	1697
16.8.2	CMPSS_REGS Registers.....	1698
16.8.3	Register to Driverlib Function Mapping.....	1723
<b>17</b>	<b>Sigma Delta Filter Module (SDFM)</b> .....	<b>1725</b>
17.1	Introduction.....	1726
17.1.1	SDFM Features.....	1726
17.1.2	Block Diagram .....	1727
17.2	Configuring Device Pins .....	1729
17.3	Input Control Unit.....	1730
17.4	Sinc Filter .....	1732
17.4.1	Data Rate and Latency of the Sinc Filter .....	1733
17.5	Data (Primary) Filter Unit.....	1734
17.5.1	32-bit or 16-bit Data Filter Output Representation .....	1734
17.5.2	Data FIFO .....	1735
17.5.3	SDSYNC Event.....	1736
17.6	Comparator (Secondary) Filter Unit.....	1738
17.6.1	Higher threshold (HLT) comparators .....	1739
17.6.2	Lower Threshold (LLT) comparators .....	1740
17.7	Interrupt Unit .....	1740
17.7.1	SDFM (SDyERR) Interrupt sources:.....	1740
17.7.2	Data Ready (DRINT) Interrupt sources:.....	1741
17.8	SDFM Registers .....	1743
17.8.1	SDFM Base Addresses .....	1743
17.8.2	SDFM_REGS Registers.....	1744
17.8.3	Register to Driverlib Function Mapping.....	1807
<b>18</b>	<b>Enhanced Pulse Width Modulator (ePWM)</b> .....	<b>1810</b>
18.1	Introduction.....	1811
18.1.1	Submodule Overview .....	1813
18.2	Configuring Device Pins .....	1818
18.3	ePWM Modules Overview.....	1818
18.4	Time-Base (TB) Submodule .....	1820
18.4.1	Purpose of the Time-Base Submodule.....	1820
18.4.2	Controlling and Monitoring the Time-Base Submodule .....	1821
18.4.3	Calculating PWM Period and Frequency .....	1822

18.4.4	Phase Locking the Time-Base Clocks of Multiple ePWM Modules.....	1826
18.4.5	Simultaneous Writes to TBPRD and CMPx Registers Between ePWM Modules .....	1826
18.4.6	Time-Base Counter Modes and Timing Waveforms.....	1826
18.4.7	Global Load .....	1829
18.5	Counter-Compare (CC) Submodule .....	1831
18.5.1	Purpose of the Counter-Compare Submodule .....	1831
18.5.2	Controlling and Monitoring the Counter-Compare Submodule.....	1831
18.5.3	Operational Highlights for the Counter-Compare Submodule.....	1832
18.5.4	Count Mode Timing Waveforms .....	1834
18.6	Action-Qualifier (AQ) Submodule .....	1837
18.6.1	Purpose of the Action-Qualifier Submodule .....	1838
18.6.2	Action-Qualifier Submodule Control and Status Register Definitions .....	1838
18.6.3	Action-Qualifier Event Priority .....	1841
18.6.4	AQCTLA and AQCTLB Shadow Mode Operations.....	1842
18.6.5	Waveforms for Common Configurations .....	1844
18.7	Dead-Band Generator (DB) Submodule .....	1851
18.7.1	Purpose of the Dead-Band Submodule .....	1851
18.7.2	Dead-band Submodule Additional Operating Modes.....	1851
18.7.3	Operational Highlights for the Dead-Band Submodule.....	1853
18.8	PWM Chopper (PC) Submodule .....	1857
18.8.1	Purpose of the PWM Chopper Submodule .....	1857
18.8.2	Operational Highlights for the PWM Chopper Submodule.....	1857
18.8.3	Waveforms .....	1858
18.9	Trip-Zone (TZ) Submodule.....	1861
18.9.1	Purpose of the Trip-Zone Submodule .....	1861
18.9.2	Operational Highlights for the Trip-Zone Submodule.....	1861
18.9.3	Generating Trip Event Interrupts .....	1864
18.10	Event-Trigger (ET) Submodule.....	1866
18.10.1	Operational Overview of the ePWM Type 4 Event-Trigger Submodule .....	1867
18.11	Digital Compare (DC) Submodule .....	1872
18.11.1	Purpose of the Digital Compare Submodule .....	1873
18.11.2	Enhanced Trip Action Using CMPSS .....	1874
18.11.3	Using CMPSS to Trip the ePWM on a Cycle-by-Cycle Basis .....	1874
18.11.4	Operation Highlights of the Digital Compare Submodule .....	1874
18.12	ePWM X-BAR .....	1880
18.13	Applications to Power Topologies .....	1884
18.13.1	Overview of Multiple Modules .....	1884
18.13.2	Key Configuration Capabilities .....	1884
18.13.3	Controlling Multiple Buck Converters With Independent Frequencies.....	1885
18.13.4	Controlling Multiple Buck Converters With Same Frequencies.....	1887
18.13.5	Controlling Multiple Half H-Bridge (HHB) Converters .....	1889
18.13.6	Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM) .....	1891
18.13.7	Practical Applications Using Phase Control Between PWM Modules .....	1894
18.13.8	Controlling a 3-Phase Interleaved DC/DC Converter .....	1895
18.13.9	Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter.....	1898
18.13.10	Controlling a Peak Current Mode Controlled Buck Module.....	1899
18.13.11	Controlling H-Bridge LLC Resonant Converter.....	1900
18.14	Register Lock Protection .....	1901
18.15	<b>High-Resolution Pulse Width Modulator (HRPWM)</b> .....	1903
18.15.1	Operational Description of HRPWM.....	1905
18.15.2	Appendix A: SFO Library Software - SFO_TI_Build_V8.lib .....	1927
18.16	ePWM Registers .....	1930
18.16.1	ePWM Base Addresses.....	1930

18.16.2	EPWM_REGS Registers .....	1931
18.16.3	SYNC_SOC_REGS Registers .....	2064
18.16.4	Register to Driverlib Function Mapping .....	2072
<b>19</b>	<b>Enhanced Capture (eCAP) .....</b>	<b>2080</b>
19.1	Introduction .....	2081
19.2	Features .....	2081
19.3	Description .....	2081
19.4	Configuring Device Pins for the eCAP .....	2082
19.5	Capture and APWM Operating Mode .....	2084
19.6	Capture Mode Description .....	2085
19.6.1	Event Prescaler .....	2086
19.6.2	Edge Polarity Select and Qualifier .....	2087
19.6.3	Continuous/One-Shot Control .....	2087
19.6.4	32-Bit Counter and Phase Control .....	2088
19.6.5	CAP1-CAP4 Registers .....	2089
19.6.6	eCAP Synchronization .....	2089
19.6.7	Interrupt Control .....	2091
19.6.8	DMA Interrupt .....	2092
19.6.9	Shadow Load and Lockout Control .....	2093
19.6.10	APWM Mode Operation .....	2093
19.7	Application of the eCAP Module .....	2094
19.7.1	Example 1 - Absolute Time-Stamp Operation Rising Edge Trigger .....	2094
19.7.2	Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger .....	2096
19.7.3	Example 3 - Time Difference (Delta) Operation Rising Edge Trigger .....	2097
19.7.4	Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger .....	2098
19.8	Application of the APWM Mode .....	2099
19.8.1	Example 1 - Simple PWM Generation (Independent Channel/s) .....	2099
19.9	eCAP Registers .....	2100
19.9.1	eCAP Base Addresses .....	2100
19.9.2	ECAP_REGS Registers .....	2101
19.9.3	Register to Driverlib Function Mapping .....	2120
<b>20</b>	<b>High Resolution Capture (HRCAP) .....</b>	<b>2122</b>
20.1	Introduction .....	2123
20.2	Description .....	2123
20.3	Operational Details .....	2123
20.3.1	HRCAP Clocking .....	2124
20.3.2	HRCAP Initialization Sequence .....	2124
20.3.3	HRCAP Interrupts .....	2126
20.3.4	HRCAP Calibration .....	2126
20.3.5	DriverLib Functions .....	2127
20.4	Known Exceptions .....	2127
20.5	HRCAP Registers .....	2128
20.5.1	HRCAP Base Addresses .....	2128
20.5.2	HRCAP_REGS Registers .....	2129
<b>21</b>	<b>Enhanced Quadrature Encoder Pulse (eQEP) .....</b>	<b>2141</b>
21.1	Introduction .....	2142
21.2	Configuring Device Pins .....	2144
21.3	Description .....	2144
21.3.1	eQEP Inputs .....	2144
21.3.2	Functional Description .....	2145
21.3.3	eQEP Memory Map .....	2146
21.4	Quadrature Decoder Unit (QDU) .....	2147
21.4.1	Position Counter Input Modes .....	2147



21.4.2	eQEP Input Polarity Selection .....	2150
21.4.3	Position-Compare Sync Output .....	2150
21.5	Position Counter and Control Unit (PCCU) .....	2150
21.5.1	Position Counter Operating Modes .....	2150
21.5.2	Position Counter Latch .....	2152
21.5.3	Position Counter Initialization .....	2154
21.5.4	eQEP Position-compare Unit .....	2155
21.6	eQEP Edge Capture Unit .....	2156
21.7	eQEP Watchdog .....	2160
21.8	Unit Timer Base .....	2160
21.9	QMA Module .....	2161
21.9.1	Modes of Operation .....	2162
21.9.2	Interrupt and Error Generation .....	2165
21.10	eQEP Interrupt Structure .....	2165
21.11	eQEP Registers .....	2167
21.11.1	eQEP Base Addresses .....	2167
21.11.2	EQEP_REGS Registers .....	2168
21.11.3	Register to Driverlib Function Mapping .....	2205
<b>22</b>	<b>Serial Peripheral Interface (SPI) .....</b>	<b>2207</b>
22.1	Introduction .....	2208
22.1.1	Features .....	2208
22.1.2	Block Diagram .....	2208
22.2	System-Level Integration .....	2209
22.2.1	SPI Module Signals .....	2209
22.2.2	Configuring Device Pins .....	2210
22.2.3	SPI Interrupts .....	2210
22.2.4	DMA Support .....	2212
22.3	SPI Operation .....	2212
22.3.1	Introduction to Operation .....	2212
22.3.2	Master Mode .....	2213
22.3.3	Slave Mode .....	2214
22.3.4	Data Format .....	2215
22.3.5	Baud Rate Selection .....	2216
22.3.6	SPI Clocking Schemes .....	2217
22.3.7	SPI FIFO Description .....	2218
22.3.8	SPI DMA Transfers .....	2219
22.3.9	SPI High-Speed Mode .....	2220
22.3.10	SPI 3-Wire Mode Description .....	2220
22.4	Programming Procedure .....	2222
22.4.1	Initialization Upon Reset .....	2222
22.4.2	Configuring the SPI .....	2222
22.4.3	Configuring the SPI for High-Speed Mode .....	2223
22.4.4	Data Transfer Example .....	2223
22.4.5	SPI 3-Wire Mode Code Examples .....	2224
22.4.6	SPI STEINV Bit in Digital Audio Transfers .....	2225
22.5	SPI Registers .....	2227
22.5.1	SPI Base Addresses .....	2227
22.5.2	SPI_REGS Registers .....	2228
22.5.3	Register to Driverlib Function Mapping .....	2247
<b>23</b>	<b>Serial Communications Interface (SCI) .....</b>	<b>2249</b>
23.1	Introduction .....	2250
23.2	Architecture .....	2252
23.3	SCI Module Signal Summary .....	2252



23.4	Configuring Device Pins .....	2252
23.5	Multiprocessor and Asynchronous Communication Modes.....	2252
23.6	SCI Programmable Data Format .....	2253
23.7	SCI Multiprocessor Communication .....	2253
23.7.1	Recognizing the Address Byte .....	2254
23.7.2	Controlling the SCI TX and RX Features .....	2254
23.7.3	Receipt Sequence.....	2254
23.8	Idle-Line Multiprocessor Mode.....	2254
23.8.1	Idle-Line Mode Steps .....	2255
23.8.2	Block Start Signal .....	2256
23.8.3	Wake-UP Temporary (WUT) Flag .....	2256
23.8.4	Receiver Operation .....	2256
23.9	Address-Bit Multiprocessor Mode .....	2256
23.9.1	Sending an Address .....	2256
23.10	SCI Communication Format .....	2257
23.10.1	Receiver Signals in Communication Modes.....	2258
23.10.2	Transmitter Signals in Communication Modes.....	2258
23.11	SCI Port Interrupts .....	2259
23.12	SCI Baud Rate Calculations .....	2260
23.13	SCI Enhanced Features.....	2260
23.13.1	SCI FIFO Description .....	2260
23.13.2	SCI Auto-Baud .....	2262
23.13.3	Autobaud-Detect Sequence .....	2262
23.14	SCI Registers.....	2263
23.14.1	SCI Base Addresses .....	2263
23.14.2	SCI_REGS Registers .....	2264
23.14.3	Register to Driverlib Function Mapping .....	2284
<b>24</b>	<b>Inter-Integrated Circuit Module (I2C).....</b>	<b>2286</b>
24.1	Introduction.....	2287
24.1.1	Features .....	2287
24.1.2	Features Not Supported.....	2288
24.1.3	Functional Overview .....	2288
24.1.4	Clock Generation.....	2289
24.1.5	I2C Clock Divider Registers (I2CCLKL and I2CCLKH) .....	2290
24.2	Configuring Device Pins .....	2290
24.3	I2C Module Operational Details .....	2291
24.3.1	Input and Output Voltage Levels .....	2291
24.3.2	Data Validity .....	2291
24.3.3	Operating Modes .....	2291
24.3.4	I2C Module START and STOP Conditions .....	2292
24.3.5	Serial Data Formats.....	2293
24.3.6	NACK Bit Generation .....	2295
24.3.7	Clock Synchronization .....	2296
24.3.8	Arbitration .....	2296
24.3.9	Digital Loopback Mode .....	2297
24.4	Interrupt Requests Generated by the I2C Module .....	2298
24.4.1	Basic I2C Interrupt Requests .....	2298
24.4.2	I2C FIFO Interrupts .....	2300
24.5	Resetting or Disabling the I2C Module .....	2301
24.6	I2C Registers .....	2302
24.6.1	I2C Base Addresses .....	2302
24.6.2	I2C_REGS Registers .....	2303
24.6.3	Register to Driverlib Function Mapping.....	2327

<b>25</b>	<b>Power Management Bus Module (PMBus)</b> .....	<b>2329</b>
25.1	Introduction .....	2330
25.1.1	Features .....	2330
25.1.2	Functional Description .....	2330
25.2	Configuring Device Pins .....	2331
25.3	Slave Mode Operation .....	2331
25.3.1	Configuration .....	2332
25.3.2	Message Handling .....	2332
25.4	Master Mode Operation .....	2340
25.4.1	Configuration .....	2341
25.4.2	Message Handling .....	2341
25.5	PMBus Registers .....	2349
25.5.1	PMBus Base Addresses .....	2349
25.5.2	PMBUS_REGS Registers .....	2350
<b>26</b>	<b>Controller Area Network (CAN)</b> .....	<b>2372</b>
26.1	Introduction .....	2373
26.1.1	Features .....	2373
26.1.2	Functional Description .....	2373
26.1.3	Block Diagram .....	2374
26.2	Configuring Device Pins .....	2375
26.3	Address/Data Bus Bridge .....	2375
26.4	Operating Modes .....	2378
26.4.1	Initialization .....	2378
26.4.2	CAN Message Transfer (Normal Operation) .....	2378
26.4.3	Test Modes .....	2379
26.5	Multiple Clock Source .....	2382
26.6	Interrupt Functionality .....	2383
26.6.1	Message Object Interrupts .....	2383
26.6.2	Status Change Interrupts .....	2383
26.6.3	Error Interrupts .....	2383
26.6.4	PIE Nomenclature for DCAN Interrupts .....	2383
26.6.5	Interrupt Topologies .....	2384
26.7	DMA Functionality .....	2385
26.8	Parity Check Mechanism .....	2385
26.8.1	Behavior on Parity Error .....	2386
26.9	Debug Mode .....	2386
26.10	Module Initialization .....	2386
26.11	Configuration of Message Objects .....	2387
26.11.1	Configuration of a Transmit Object for Data Frames .....	2387
26.11.2	Configuration of a Transmit Object for Remote Frames .....	2387
26.11.3	Configuration of a Single Receive Object for Data Frames .....	2387
26.11.4	Configuration of a Single Receive Object for Remote Frames .....	2388
26.11.5	Configuration of a FIFO Buffer .....	2388
26.12	Message Handling .....	2388
26.12.1	Message Handler Overview .....	2388
26.12.2	Receive/Transmit Priority .....	2389
26.12.3	Transmission of Messages in Event Driven CAN Communication .....	2389
26.12.4	Updating a Transmit Object .....	2390
26.12.5	Changing a Transmit Object .....	2390
26.12.6	Acceptance Filtering of Received Messages .....	2390
26.12.7	Reception of Data Frames .....	2390
26.12.8	Reception of Remote Frames .....	2391
26.12.9	Reading Received Messages .....	2391

26.12.10	Requesting New Data for a Receive Object .....	2391
26.12.11	Storing Received Messages in FIFO Buffers .....	2391
26.12.12	Reading from a FIFO Buffer .....	2392
26.13	CAN Bit Timing .....	2393
26.13.1	Bit Time and Bit Rate .....	2394
26.13.2	Configuration of the CAN Bit Timing .....	2399
26.14	Message Interface Register Sets .....	2401
26.14.1	Message Interface Register Sets 1 and 2 .....	2402
26.14.2	IF3 Register Set .....	2403
26.15	Message RAM .....	2404
26.15.1	Structure of Message Objects .....	2404
26.15.2	Addressing Message Objects in RAM .....	2406
26.15.3	Message RAM Representation in Debug Mode .....	2407
26.16	CAN Registers.....	2408
26.16.1	CAN Base Addresses .....	2408
26.16.2	CAN_REGS Registers .....	2409
26.16.3	Register to Driverlib Function Mapping .....	2470
<b>27</b>	<b>Local Interconnect Network (LIN) .....</b>	<b>2472</b>
27.1	Introduction.....	2473
27.1.1	SCI Features.....	2473
27.1.2	LIN Features .....	2474
27.1.3	Block Diagram .....	2475
27.2	Serial Communications Interface Module .....	2478
27.2.1	SCI Communication Formats .....	2478
27.2.2	SCI Interrupts .....	2487
27.2.3	SCI DMA Interface .....	2491
27.2.4	SCI Configurations .....	2492
27.2.5	SCI Low-Power Mode .....	2494
27.3	Local Interconnect Network Module .....	2495
27.3.1	LIN Communication Formats .....	2495
27.3.2	LIN Interrupts .....	2512
27.3.3	Servicing LIN interrupts.....	2512
27.3.4	LIN DMA Interface .....	2512
27.3.5	LIN Configurations .....	2514
27.4	Low-Power Mode.....	2516
27.4.1	Entering Sleep Mode .....	2516
27.4.2	Wakeup.....	2517
27.4.3	Wakeup Timeouts.....	2518
27.5	Emulation Mode .....	2518
27.6	SCI/LIN Registers .....	2519
27.6.1	LIN Base Addresses .....	2519
27.6.2	LIN_REGS Registers .....	2520
<b>28</b>	<b>Fast Serial Interface (FSI) .....</b>	<b>2579</b>
28.1	Introduction.....	2580
28.2	Features .....	2580
28.3	System-level Integration.....	2580
28.3.1	CPU Interface.....	2580
28.3.2	Signal Description .....	2582
28.3.3	Interrupts .....	2583
28.3.4	CLA Task Triggering.....	2585
28.3.5	DMA Interface .....	2585
28.3.6	External Frame Trigger Mux .....	2586
28.4	FSI Operation .....	2586

28.4.1	Introduction to Operation .....	2586
28.4.2	FSI Transmitter Module .....	2586
28.4.3	FSI Receiver Module .....	2592
28.4.4	Frame Format.....	2598
28.4.5	The Flush Sequence.....	2601
28.4.6	Internal Loopback .....	2602
28.4.7	CRC Generation.....	2602
28.4.8	ECC Module .....	2603
28.4.9	SPI Compatibility Mode.....	2603
28.5	Programmer's Model.....	2607
28.5.1	Establishing the Communication Link .....	2607
28.5.2	Register Protection.....	2609
28.5.3	Emulation Mode .....	2609
28.6	FSI Registers .....	2610
28.6.1	FSI Base Addresses .....	2610
28.6.2	FSI_RX_REGS Registers .....	2611
28.6.3	FSI_TX_REGS Registers .....	2652
28.7	Register to Driverlib Function Mapping.....	2680
<b>29</b>	<b>Configurable Logic Block (CLB).....</b>	<b>2684</b>
29.1	Introduction.....	2685
29.2	Features .....	2685
29.3	CLB Input/Output Connection .....	2686
29.3.1	Overview .....	2686
29.3.2	CLB Input Selection.....	2686
29.3.3	CLB Output Selection.....	2690
29.3.4	Peripheral Signal Multiplexer .....	2691
29.4	The CLB Tile.....	2693
29.4.1	Static Switch Block .....	2694
29.4.2	Counter Block.....	2696
29.4.3	FSM Block.....	2698
29.4.4	LUT4 Block.....	2699
29.4.5	Output LUT Block .....	2699
29.4.6	High Level Controller (HLC) .....	2700
29.5	CPU Interface.....	2703
29.5.1	Register Description .....	2703
29.5.2	Non-Memory Mapped Registers .....	2704
29.6	CLB Registers .....	2705
29.6.1	CLB Base Addresses .....	2705
29.6.2	CLB_LOGIC_CONFIG_REGS Registers .....	2706
29.6.3	CLB_LOGIC_CONTROL_REGS Registers.....	2739
29.6.4	CLB_DATA_EXCHANGE_REGS Registers.....	2766
29.6.5	Register to Driverlib Function Mapping.....	2769
	<b>Revision History .....</b>	<b>2772</b>

## List of Figures

3-1.	Device Interrupt Architecture .....	91
3-2.	Interrupt Propagation Path .....	92
3-3.	Clocking System .....	104
3-4.	System PLL.....	104
3-5.	Using GPIO18 when INTOSC2 is the SYSCLK source .....	105
3-6.	Single-ended 3.3V External Clock.....	106
3-7.	External Crystal .....	106
3-8.	External Resonator .....	107
3-9.	CPU Timers.....	114
3-10.	CPU Timer Interrupt Signals and Output Signal .....	114
3-11.	Watchdog Timer Module .....	115
3-12.	Memory Architecture .....	121
3-13.	Arbitration Scheme on Global Shared Memories.....	123
3-14.	Arbitration Scheme on Local Shared Memories .....	123
3-15.	FMC Interface with Core, Bank and Pump .....	129
3-16.	Flash Prefetch Mode .....	132
3-17.	ECC Logic Inputs and Outputs.....	135
3-18.	Storage of Zone-Select Bits in OTP .....	144
3-19.	Location of Zone-Select Block Based on Link-Pointer for Bank0 .....	145
3-20.	Location of Zone-Select Block Based on Link-Pointer for Bank1 .....	146
3-21.	CSM Password Match Flow (PMF).....	151
3-22.	ECSS Password Match Flow (PMF).....	153
3-23.	NMAVFLG Register .....	158
3-24.	NMAVSET Register .....	160
3-25.	NMAVCLR Register .....	162
3-26.	NMAVINTEN Register.....	164
3-27.	NMCPURDAVADDR Register .....	165
3-28.	NMCPUWRAVADDR Register.....	166
3-29.	NMCPUFAVADDR Register.....	167
3-30.	NMDMAWRAVADDR Register .....	168
3-31.	NMCLA1RDAVADDR Register .....	169
3-32.	NMCLA1WRAVADDR Register.....	170
3-33.	NMCLA1FAVADDR Register.....	171
3-34.	MAVFLG Register .....	172
3-35.	MAVSET Register .....	173
3-36.	MAVCLR Register .....	174
3-37.	MAVINTEN Register .....	175
3-38.	MCPUFAVADDR Register.....	176
3-39.	MCPUWRAVADDR Register.....	177
3-40.	MDMAWRAVADDR Register .....	178
3-41.	CLKCFGLOCK1 Register.....	180
3-42.	CLKSRCCTL1 Register .....	182
3-43.	CLKSRCCTL2 Register .....	184
3-44.	CLKSRCCTL3 Register .....	185
3-45.	SYSPLLCTL1 Register.....	186
3-46.	SYSPLLMULT Register .....	187
3-47.	SYSPLLSTS Register .....	188

3-48.	SYSCLKDIVSEL Register .....	189
3-49.	XCLKOUTDIVSEL Register .....	190
3-50.	LOSPCP Register .....	191
3-51.	MCDCCR Register .....	192
3-52.	X1CNT Register.....	193
3-53.	XTALCR Register.....	194
3-54.	CPUSYSLOCK1 Register .....	197
3-55.	PIEVERRADDR Register.....	200
3-56.	PCLKCR0 Register.....	201
3-57.	PCLKCR2 Register .....	203
3-58.	PCLKCR3 Register.....	205
3-59.	PCLKCR4 Register.....	207
3-60.	PCLKCR6 Register.....	208
3-61.	PCLKCR7 Register.....	209
3-62.	PCLKCR8 Register.....	210
3-63.	PCLKCR9 Register.....	211
3-64.	PCLKCR10 Register .....	212
3-65.	PCLKCR13 Register .....	213
3-66.	PCLKCR14 Register .....	214
3-67.	PCLKCR15 Register .....	216
3-68.	PCLKCR16 Register .....	218
3-69.	PCLKCR17 Register .....	219
3-70.	PCLKCR18 Register .....	220
3-71.	PCLKCR19 Register .....	221
3-72.	PCLKCR20 Register .....	222
3-73.	PCLKCR21 Register .....	223
3-74.	LPMCR Register .....	224
3-75.	GPIOLPMSEL0 Register.....	225
3-76.	GPIOLPMSEL1 Register.....	228
3-77.	TMR2CLKCTL Register.....	231
3-78.	RESCCLR Register.....	232
3-79.	RESC Register .....	234
3-80.	TIM Register.....	237
3-81.	PRD Register.....	238
3-82.	TCR Register .....	239
3-83.	TPR Register .....	241
3-84.	TPRH Register .....	242
3-85.	B0_Z1_LINKPOINTER Register .....	244
3-86.	Z1_OTPSECLOCK Register .....	245
3-87.	Z1_BOOTDEF_HIGH Register .....	246
3-88.	B0_Z1_LINKPOINTERERR Register .....	247
3-89.	Z1_BOOTPIN_CONFIG Register.....	248
3-90.	Z1_GPREG2 Register.....	249
3-91.	Z1_BOOTDEF_LOW Register .....	250
3-92.	Z1_CSMKEY0 Register .....	251
3-93.	Z1_CSMKEY1 Register .....	252
3-94.	Z1_CSMKEY2 Register .....	253
3-95.	Z1_CSMKEY3 Register .....	254
3-96.	Z1_CR Register .....	255

3-97.	B0_Z1_GRABSECTR Register .....	256
3-98.	Z1_GRABRAMR Register .....	260
3-99.	B0_Z1_EXEONLYSECTR Register .....	262
3-100.	Z1_EXEONLYRAMR Register .....	266
3-101.	B0_Z2_LINKPOINTER Register .....	269
3-102.	Z2_OTPSECLOCK Register .....	270
3-103.	B0_Z2_LINKPOINTERERR Register .....	271
3-104.	Z2_CSMKEY0 Register .....	272
3-105.	Z2_CSMKEY1 Register .....	273
3-106.	Z2_CSMKEY2 Register .....	274
3-107.	Z2_CSMKEY3 Register .....	275
3-108.	Z2_CR Register .....	276
3-109.	B0_Z2_GRABSECTR Register .....	277
3-110.	Z2_GRABRAMR Register .....	281
3-111.	B0_Z2_EXEONLYSECTR Register .....	283
3-112.	Z2_EXEONLYRAMR Register .....	287
3-113.	B1_Z1_LINKPOINTER Register .....	290
3-114.	B1_Z1_LINKPOINTERERR Register .....	291
3-115.	B1_Z1_GRABSECTR Register .....	292
3-116.	B1_Z1_EXEONLYSECTR Register .....	296
3-117.	B1_Z2_LINKPOINTER Register .....	301
3-118.	B1_Z2_LINKPOINTERERR Register .....	302
3-119.	B1_Z2_GRABSECTR Register .....	303
3-120.	B1_Z2_EXEONLYSECTR Register .....	307
3-121.	FLSEM Register .....	312
3-122.	B0_SECTSTAT Register.....	313
3-123.	RAMSTAT Register.....	316
3-124.	B1_SECTSTAT Register.....	318
3-125.	SECERRSTAT Register .....	321
3-126.	SECERRCLR Register .....	322
3-127.	SECERRFRC Register.....	323
3-128.	PARTIDL Register .....	326
3-129.	PARTIDH Register.....	328
3-130.	REVID Register .....	329
3-131.	DC21 Register.....	330
3-132.	FUSEERR Register.....	331
3-133.	SOFTPRES0 Register .....	332
3-134.	SOFTPRES2 Register .....	333
3-135.	SOFTPRES3 Register .....	335
3-136.	SOFTPRES4 Register .....	336
3-137.	SOFTPRES6 Register .....	337
3-138.	SOFTPRES7 Register .....	338
3-139.	SOFTPRES8 Register .....	339
3-140.	SOFTPRES9 Register .....	340
3-141.	SOFTPRES10 Register .....	341
3-142.	SOFTPRES13 Register .....	342
3-143.	SOFTPRES14 Register .....	343
3-144.	SOFTPRES15 Register .....	344
3-145.	SOFTPRES16 Register .....	345

3-146. SOFTPRES17 Register .....	346
3-147. SOFTPRES18 Register .....	347
3-148. SOFTPRES19 Register .....	348
3-149. SOFTPRES20 Register .....	349
3-150. SOFTPRES21 Register .....	350
3-151. TAP_STATUS Register .....	351
3-152. CLA1TASKSRCSELLOCK Register .....	353
3-153. DMACHSRCSELLOCK Register .....	354
3-154. CLA1TASKSRCSEL1 Register .....	355
3-155. CLA1TASKSRCSEL2 Register .....	356
3-156. DMACHSRCSEL1 Register .....	357
3-157. DMACHSRCSEL2 Register .....	358
3-158. FRDCNTL Register .....	360
3-159. FBAC Register .....	361
3-160. FBFALLBACK Register .....	362
3-161. FBPRDY Register .....	363
3-162. FPAC1 Register .....	364
3-163. FPAC2 Register .....	365
3-164. FMSTAT Register .....	366
3-165. FRD_INTF_CTRL Register .....	368
3-166. ECC_ENABLE Register .....	371
3-167. SINGLE_ERR_ADDR_LOW Register .....	372
3-168. SINGLE_ERR_ADDR_HIGH Register .....	373
3-169. UNC_ERR_ADDR_LOW Register .....	374
3-170. UNC_ERR_ADDR_HIGH Register .....	375
3-171. ERR_STATUS Register .....	376
3-172. ERR_POS Register .....	378
3-173. ERR_STATUS_CLR Register .....	379
3-174. ERR_CNT Register .....	380
3-175. ERR_THRESHOLD Register .....	381
3-176. ERR_INTFLG Register .....	382
3-177. ERR_INTCLR Register .....	383
3-178. FDATAH_TEST Register .....	384
3-179. FDATAI_TEST Register .....	385
3-180. FADDR_TEST Register .....	386
3-181. FECC_TEST Register .....	387
3-182. FECC_CTRL Register .....	388
3-183. FOUTH_TEST Register .....	389
3-184. FOUTL_TEST Register .....	390
3-185. FECC_STATUS Register .....	391
3-186. DxLOCK Register .....	394
3-187. DxCOMMIT Register .....	395
3-188. DxACCPROT0 Register .....	396
3-189. DxTEST Register .....	397
3-190. DxINIT Register .....	398
3-191. DxINITDONE Register .....	399
3-192. LSxLOCK Register .....	400
3-193. LSxCOMMIT Register .....	402
3-194. LSxMSEL Register .....	404



3-195. LSxCLAPGM Register .....	406
3-196. LSxACCPROT0 Register .....	408
3-197. LSxACCPROT1 Register .....	410
3-198. LSxTEST Register .....	412
3-199. LSxINIT Register .....	414
3-200. LSxINITDONE Register .....	416
3-201. GSxLOCK Register .....	418
3-202. GSxCOMMIT Register .....	420
3-203. GSxACCPROT0 Register .....	422
3-204. GSxACCPROT1 Register .....	424
3-205. GSxACCPROT2 Register .....	425
3-206. GSxACCPROT3 Register .....	426
3-207. GSxTEST Register .....	427
3-208. GSxINIT Register .....	429
3-209. GSxINITDONE Register .....	431
3-210. MSGxLOCK Register .....	433
3-211. MSGxCOMMIT Register .....	434
3-212. MSGxTEST Register .....	435
3-213. MSGxINIT Register .....	436
3-214. MSGxINITDONE Register .....	437
3-215. UCERRFLG Register .....	439
3-216. UCERRSET Register .....	440
3-217. UCERRCLR Register .....	441
3-218. UCCPUREADDR Register .....	442
3-219. UCDMAREADDR Register .....	443
3-220. UCCLA1READDR Register .....	444
3-221. CERRFLG Register .....	445
3-222. CERRSET Register .....	446
3-223. CERRCLR Register .....	447
3-224. CCPUREADDR Register .....	448
3-225. CERRCNT Register .....	449
3-226. CERRTHRES Register .....	450
3-227. CEINTFLG Register .....	451
3-228. CEINTCLR Register .....	452
3-229. CEINTSET Register .....	453
3-230. CEINTEN Register .....	454
3-231. NMICFG Register .....	456
3-232. NMIFLG Register .....	457
3-233. NMIFLGCLR Register .....	459
3-234. NMIFLGFRC Register .....	460
3-235. NMIWDCNT Register .....	461
3-236. NMIWDPRD Register .....	462
3-237. NMISHDFLG Register .....	463
3-238. ADCA_AC Register .....	466
3-239. ADCB_AC Register .....	467
3-240. ADCC_AC Register .....	468
3-241. CMPSS1_AC Register .....	469
3-242. CMPSS2_AC Register .....	470
3-243. CMPSS3_AC Register .....	471

3-244. CMPSS4_AC Register .....	472
3-245. CMPSS5_AC Register .....	473
3-246. CMPSS6_AC Register .....	474
3-247. CMPSS7_AC Register .....	475
3-248. DACA_AC Register .....	476
3-249. DACB_AC Register .....	477
3-250. PGA1_AC Register .....	478
3-251. PGA2_AC Register .....	479
3-252. PGA3_AC Register .....	480
3-253. PGA4_AC Register .....	481
3-254. PGA5_AC Register .....	482
3-255. PGA6_AC Register .....	483
3-256. PGA7_AC Register .....	484
3-257. EPWM1_AC Register .....	485
3-258. EPWM2_AC Register .....	486
3-259. EPWM3_AC Register .....	487
3-260. EPWM4_AC Register .....	488
3-261. EPWM5_AC Register .....	489
3-262. EPWM6_AC Register .....	490
3-263. EPWM7_AC Register .....	491
3-264. EPWM8_AC Register .....	492
3-265. EQEP1_AC Register .....	493
3-266. EQEP2_AC Register .....	494
3-267. ECAP1_AC Register .....	495
3-268. ECAP2_AC Register .....	496
3-269. ECAP3_AC Register .....	497
3-270. ECAP4_AC Register .....	498
3-271. ECAP5_AC Register .....	499
3-272. ECAP6_AC Register .....	500
3-273. ECAP7_AC Register .....	501
3-274. SDFM1_AC Register .....	502
3-275. CLB1_AC Register .....	503
3-276. CLB2_AC Register .....	504
3-277. CLB3_AC Register .....	505
3-278. CLB4_AC Register .....	506
3-279. CLA1PROMCRC_AC Register .....	507
3-280. SPIA_AC Register .....	508
3-281. SPIB_AC Register .....	509
3-282. PMBUS_A_AC Register .....	510
3-283. LIN_A_AC Register .....	511
3-284. DCANA_AC Register .....	512
3-285. DCANB_AC Register .....	513
3-286. FSIATX_AC Register .....	514
3-287. FSIARX_AC Register .....	515
3-288. HRPWM_A_AC Register .....	516
3-289. PERIPH_AC_LOCK Register .....	517
3-290. PIECTRL Register .....	520
3-291. PIEACK Register .....	521
3-292. PIEIER1 Register .....	522

3-293. PIEIFR1 Register .....	524
3-294. PIEIER2 Register .....	526
3-295. PIEIFR2 Register .....	528
3-296. PIEIER3 Register .....	530
3-297. PIEIFR3 Register .....	532
3-298. PIEIER4 Register .....	534
3-299. PIEIFR4 Register .....	536
3-300. PIEIER5 Register .....	538
3-301. PIEIFR5 Register .....	540
3-302. PIEIER6 Register .....	542
3-303. PIEIFR6 Register .....	544
3-304. PIEIER7 Register .....	546
3-305. PIEIFR7 Register .....	548
3-306. PIEIER8 Register .....	550
3-307. PIEIFR8 Register .....	552
3-308. PIEIER9 Register .....	554
3-309. PIEIFR9 Register .....	556
3-310. PIEIER10 Register.....	558
3-311. PIEIFR10 Register.....	560
3-312. PIEIER11 Register.....	562
3-313. PIEIFR11 Register.....	564
3-314. PIEIER12 Register.....	566
3-315. PIEIFR12 Register.....	568
3-316. UID_PSRAND0 Register.....	573
3-317. UID_PSRAND1 Register.....	574
3-318. UID_PSRAND2 Register.....	575
3-319. UID_PSRAND3 Register.....	576
3-320. UID_PSRAND4 Register.....	577
3-321. UID_PSRAND5 Register.....	578
3-322. UID_UNIQUE Register .....	579
3-323. UID_CHECKSUM Register .....	580
3-324. SCSR Register .....	582
3-325. WDCNTR Register .....	583
3-326. WDKEY Register.....	584
3-327. WDCR Register .....	585
3-328. WDWCR Register .....	587
3-329. XINT1CR Register .....	589
3-330. XINT2CR Register.....	590
3-331. XINT3CR Register.....	591
3-332. XINT4CR Register .....	592
3-333. XINT5CR Register .....	593
3-334. XINT1CTR Register .....	594
3-335. XINT2CTR Register .....	595
3-336. XINT3CTR Register .....	596
4-1. Device Boot Flow .....	614
4-2. Emulation Boot Flow .....	615
4-3. Standalone Boot Flow.....	616
4-4. Overview of SCI Bootloader Operation .....	622
4-5. Overview of SCI Boot Function .....	623

4-6.	SPI Loader .....	624
4-7.	Data Transfer From EEPROM Flow .....	625
4-8.	EEPROM Device at Address 0x50 .....	626
4-9.	Overview of I2C Boot Function .....	627
4-10.	Random Read .....	628
4-11.	Sequential Read .....	628
4-12.	Overview of Parallel GPIO Bootloader Operation .....	629
4-13.	Parallel GPIO Bootloader Handshake Protocol .....	630
4-14.	Parallel GPIO Mode Overview .....	630
4-15.	Parallel GPIO Mode - Host Transfer Flow .....	631
4-16.	8-Bit Parallel GetWord Function .....	632
4-17.	Overview of CAN-A Bootloader Operation .....	633
5-1.	CLA (Type 2) Block Diagram .....	644
5-2.	_MVECTBGRNDACTIVE Register .....	783
5-3.	_MPSACTL Register .....	784
5-4.	_MPSA1 Register .....	786
5-5.	_MPSA2 Register .....	787
5-6.	SOFTINTEN Register .....	788
5-7.	SOFTINTFRC Register .....	790
5-8.	SOFTINTEN Register .....	792
5-9.	SOFTINTFRC Register .....	794
5-10.	MVECT1 Register .....	797
5-11.	MVECT2 Register .....	798
5-12.	MVECT3 Register .....	799
5-13.	MVECT4 Register .....	800
5-14.	MVECT5 Register .....	801
5-15.	MVECT6 Register .....	802
5-16.	MVECT7 Register .....	803
5-17.	MVECT8 Register .....	804
5-18.	MCTL Register .....	805
5-19.	_MVECTBGRNDACTIVE Register .....	806
5-20.	SOFTINTEN Register .....	807
5-21.	_MSTSBGRND Register .....	809
5-22.	_MCTLBGRND Register .....	810
5-23.	_MVECTBGRND Register .....	811
5-24.	MIFR Register .....	812
5-25.	MIOVF Register .....	817
5-26.	MIFRC Register .....	820
5-27.	MICLR Register .....	822
5-28.	MICLROVF Register .....	824
5-29.	MIER Register .....	826
5-30.	MIRUN Register .....	829
5-31.	_MPC Register .....	832
5-32.	_MAR0 Register .....	833
5-33.	_MAR1 Register .....	834
5-34.	_MSTF Register .....	835
5-35.	_MR0 Register .....	838
5-36.	_MR1 Register .....	839
5-37.	_MR2 Register .....	840

5-38.	_MR3 Register .....	841
5-39.	_MPSACTL Register .....	842
5-40.	_MPSA1 Register .....	844
5-41.	_MPSA2 Register .....	845
6-1.	DCC Operation .....	847
6-2.	DCCGCTRL Register .....	852
6-3.	DCCREV Register .....	853
6-4.	DCCNTSEED0 Register .....	854
6-5.	DCCVALIDSEED0 Register .....	855
6-6.	DCCNTSEED1 Register .....	856
6-7.	DCCSTATUS Register .....	857
6-8.	DCCNT0 Register .....	858
6-9.	DCCVALID0 Register .....	859
6-10.	DCCNT1 Register .....	860
6-11.	DCCCLKSRC1 Register .....	861
6-12.	DCCCLKSRC0 Register .....	862
7-1.	CLAPROMCRC Functional Diagram .....	865
7-2.	CRC32_CONTROLREG Register .....	869
7-3.	CRC32_STARTADDRESS Register .....	871
7-4.	CRC32_SEED Register .....	872
7-5.	CRC32_STATUSREG Register .....	873
7-6.	CRC32_CRCRESULT Register .....	874
7-7.	CRC32_GOLDENCRC Register .....	875
7-8.	CRC32_INTEN Register .....	876
7-9.	CRC32_FLG Register .....	877
7-10.	CRC32_CLR Register .....	878
7-11.	CRC32_FRC Register .....	879
8-1.	GPIO Logic for a Single Pin .....	881
8-2.	Input Qualification Using a Sampling Window .....	885
8-3.	Input Qualifier Clock Cycles .....	887
8-4.	GPACTRL Register .....	897
8-5.	GPAQSEL1 Register .....	898
8-6.	GPAQSEL2 Register .....	900
8-7.	GPAMUX1 Register .....	902
8-8.	GPAMUX2 Register .....	904
8-9.	GPADIR Register .....	906
8-10.	GPAPUD Register .....	908
8-11.	GPAINV Register .....	910
8-12.	GPAODR Register .....	912
8-13.	GPAAMSEL Register .....	914
8-14.	GPAGMUX1 Register .....	916
8-15.	GPAGMUX2 Register .....	918
8-16.	GPACSEL1 Register .....	920
8-17.	GPACSEL2 Register .....	921
8-18.	GPACSEL3 Register .....	922
8-19.	GPACSEL4 Register .....	923
8-20.	GPALOCK Register .....	924
8-21.	GPACR Register .....	926
8-22.	GPBCTRL Register .....	928

8-23.	GPBQSEL1 Register .....	929
8-24.	GPBQSEL2 Register .....	931
8-25.	GPBMUX1 Register .....	932
8-26.	GPBMUX2 Register .....	933
8-27.	GPBDIR Register .....	934
8-28.	GPBPUD Register .....	936
8-29.	GPBINV Register .....	938
8-30.	GPBODR Register.....	940
8-31.	GPBGMUX1 Register .....	942
8-32.	GPBGMUX2 Register .....	943
8-33.	GPBCSEL1 Register .....	944
8-34.	GPBCSEL2 Register .....	945
8-35.	GPBCSEL3 Register .....	946
8-36.	GPBCSEL4 Register .....	947
8-37.	GPBLOCK Register .....	948
8-38.	GPBCR Register.....	950
8-39.	GPHCTRL Register.....	952
8-40.	GPHQSEL1 Register .....	953
8-41.	GPHQSEL2 Register .....	955
8-42.	GPHPUD Register .....	957
8-43.	GPHINV Register .....	959
8-44.	GPHAMSEL Register .....	961
8-45.	GPHLOCK Register .....	963
8-46.	GPHCR Register.....	965
8-47.	GPADAT Register .....	968
8-48.	GPASET Register .....	970
8-49.	GPACLEAR Register.....	972
8-50.	GPATOGGLE Register .....	974
8-51.	GPBDAT Register .....	976
8-52.	GPBSET Register .....	978
8-53.	GPBCLEAR Register.....	980
8-54.	GPBTOGGLE Register .....	982
8-55.	GPHDAT Register .....	984
9-1.	Input X-BAR .....	990
9-2.	ePWM X-BAR Architecture - Single Output .....	992
9-3.	CLB X-BAR Architecture - Single Output .....	994
9-4.	GPIO Output X-BAR Architecture .....	996
9-5.	ePWM and Output X-BARs Sources .....	998
9-6.	XBARFLG1 Register.....	1001
9-7.	XBARFLG2 Register.....	1007
9-8.	XBARFLG3 Register.....	1013
9-9.	XBARFLG4 Register.....	1019
9-10.	XBARCLR1 Register.....	1024
9-11.	XBARCLR2 Register.....	1027
9-12.	XBARCLR3 Register.....	1030
9-13.	XBARCLR4 Register.....	1033
9-14.	INPUT1SELECT Register.....	1037
9-15.	INPUT2SELECT Register.....	1038
9-16.	INPUT3SELECT Register.....	1039

9-17.	INPUT4SELECT Register .....	1040
9-18.	INPUT5SELECT Register .....	1041
9-19.	INPUT6SELECT Register .....	1042
9-20.	INPUT7SELECT Register .....	1043
9-21.	INPUT8SELECT Register .....	1044
9-22.	INPUT9SELECT Register .....	1045
9-23.	INPUT10SELECT Register .....	1046
9-24.	INPUT11SELECT Register .....	1047
9-25.	INPUT12SELECT Register .....	1048
9-26.	INPUT13SELECT Register .....	1049
9-27.	INPUT14SELECT Register .....	1050
9-28.	INPUT15SELECT Register .....	1051
9-29.	INPUT16SELECT Register .....	1052
9-30.	INPUTSELECTLOCK Register .....	1053
9-31.	OUTPUT1MUX0TO15CFG Register .....	1057
9-32.	OUTPUT1MUX16TO31CFG Register .....	1060
9-33.	OUTPUT2MUX0TO15CFG Register .....	1063
9-34.	OUTPUT2MUX16TO31CFG Register .....	1066
9-35.	OUTPUT3MUX0TO15CFG Register .....	1069
9-36.	OUTPUT3MUX16TO31CFG Register .....	1072
9-37.	OUTPUT4MUX0TO15CFG Register .....	1075
9-38.	OUTPUT4MUX16TO31CFG Register .....	1078
9-39.	OUTPUT5MUX0TO15CFG Register .....	1081
9-40.	OUTPUT5MUX16TO31CFG Register .....	1084
9-41.	OUTPUT6MUX0TO15CFG Register .....	1087
9-42.	OUTPUT6MUX16TO31CFG Register .....	1090
9-43.	OUTPUT7MUX0TO15CFG Register .....	1093
9-44.	OUTPUT7MUX16TO31CFG Register .....	1096
9-45.	OUTPUT8MUX0TO15CFG Register .....	1099
9-46.	OUTPUT8MUX16TO31CFG Register .....	1102
9-47.	OUTPUT1MUXENABLE Register .....	1105
9-48.	OUTPUT2MUXENABLE Register .....	1110
9-49.	OUTPUT3MUXENABLE Register .....	1115
9-50.	OUTPUT4MUXENABLE Register .....	1120
9-51.	OUTPUT5MUXENABLE Register .....	1125
9-52.	OUTPUT6MUXENABLE Register .....	1130
9-53.	OUTPUT7MUXENABLE Register .....	1135
9-54.	OUTPUT8MUXENABLE Register .....	1140
9-55.	OUTPUTLATCH Register .....	1145
9-56.	OUTPUTLATCHCLR Register .....	1147
9-57.	OUTPUTLATCHFRC Register .....	1149
9-58.	OUTPUTLATCHENABLE Register .....	1151
9-59.	OUTPUTINV Register .....	1153
9-60.	OUTPUTLOCK Register .....	1155
9-61.	TRIP4MUX0TO15CFG Register .....	1158
9-62.	TRIP4MUX16TO31CFG Register .....	1161
9-63.	TRIP5MUX0TO15CFG Register .....	1164
9-64.	TRIP5MUX16TO31CFG Register .....	1167
9-65.	TRIP7MUX0TO15CFG Register .....	1170



9-66.	TRIP7MUX16TO31CFG Register .....	1173
9-67.	TRIP8MUX0TO15CFG Register .....	1176
9-68.	TRIP8MUX16TO31CFG Register .....	1179
9-69.	TRIP9MUX0TO15CFG Register .....	1182
9-70.	TRIP9MUX16TO31CFG Register .....	1185
9-71.	TRIP10MUX0TO15CFG Register .....	1188
9-72.	TRIP10MUX16TO31CFG Register .....	1191
9-73.	TRIP11MUX0TO15CFG Register .....	1194
9-74.	TRIP11MUX16TO31CFG Register .....	1197
9-75.	TRIP12MUX0TO15CFG Register .....	1200
9-76.	TRIP12MUX16TO31CFG Register .....	1203
9-77.	TRIP4MUXENABLE Register.....	1206
9-78.	TRIP5MUXENABLE Register.....	1211
9-79.	TRIP7MUXENABLE Register.....	1216
9-80.	TRIP8MUXENABLE Register.....	1221
9-81.	TRIP9MUXENABLE Register.....	1226
9-82.	TRIP10MUXENABLE Register .....	1231
9-83.	TRIP11MUXENABLE Register .....	1236
9-84.	TRIP12MUXENABLE Register .....	1241
9-85.	TRIPOUTINV Register.....	1246
9-86.	TRIPLOCK Register .....	1248
9-87.	AUXSIG0MUX0TO15CFG Register .....	1251
9-88.	AUXSIG0MUX16TO31CFG Register.....	1255
9-89.	AUXSIG1MUX0TO15CFG Register .....	1259
9-90.	AUXSIG1MUX16TO31CFG Register.....	1263
9-91.	AUXSIG2MUX0TO15CFG Register .....	1267
9-92.	AUXSIG2MUX16TO31CFG Register.....	1271
9-93.	AUXSIG3MUX0TO15CFG Register .....	1275
9-94.	AUXSIG3MUX16TO31CFG Register.....	1279
9-95.	AUXSIG4MUX0TO15CFG Register .....	1283
9-96.	AUXSIG4MUX16TO31CFG Register.....	1287
9-97.	AUXSIG5MUX0TO15CFG Register .....	1291
9-98.	AUXSIG5MUX16TO31CFG Register.....	1295
9-99.	AUXSIG6MUX0TO15CFG Register .....	1299
9-100.	AUXSIG6MUX16TO31CFG Register.....	1303
9-101.	AUXSIG7MUX0TO15CFG Register .....	1307
9-102.	AUXSIG7MUX16TO31CFG Register.....	1311
9-103.	AUXSIG0MUXENABLE Register.....	1315
9-104.	AUXSIG1MUXENABLE Register.....	1320
9-105.	AUXSIG2MUXENABLE Register.....	1325
9-106.	AUXSIG3MUXENABLE Register.....	1330
9-107.	AUXSIG4MUXENABLE Register.....	1335
9-108.	AUXSIG5MUXENABLE Register.....	1340
9-109.	AUXSIG6MUXENABLE Register.....	1345
9-110.	AUXSIG7MUXENABLE Register.....	1350
9-111.	AUXSIGOUTINV Register.....	1355
9-112.	AUXSIGLOCK Register .....	1357
10-1.	DMA Block Diagram .....	1365
10-2.	DMA Trigger Architecture .....	1366



10-3. Peripheral Interrupt Trigger Input Diagram .....	1367
10-4. DMA State Diagram.....	1373
10-5. 3-Stage Pipeline DMA Transfer .....	1375
10-6. 3-stage Pipeline With One Read Stall .....	1375
10-7. Overrun Detection Logic.....	1377
10-8. DMACTRL Register .....	1380
10-9. DEBUGCTRL Register .....	1381
10-10. PRIORITYCTRL1 Register.....	1382
10-11. PRIORITYSTAT Register .....	1383
10-12. MODE Register .....	1386
10-13. CONTROL Register.....	1388
10-14. BURST_SIZE Register .....	1390
10-15. BURST_COUNT Register.....	1391
10-16. SRC_BURST_STEP Register .....	1392
10-17. DST_BURST_STEP Register .....	1393
10-18. TRANSFER_SIZE Register .....	1394
10-19. TRANSFER_COUNT Register .....	1395
10-20. SRC_TRANSFER_STEP Register.....	1396
10-21. DST_TRANSFER_STEP Register .....	1397
10-22. SRC_WRAP_SIZE Register .....	1398
10-23. SRC_WRAP_COUNT Register.....	1399
10-24. SRC_WRAP_STEP Register .....	1400
10-25. DST_WRAP_SIZE Register .....	1401
10-26. DST_WRAP_COUNT Register.....	1402
10-27. DST_WRAP_STEP Register .....	1403
10-28. SRC_BEG_ADDR_SHADOW Register .....	1404
10-29. SRC_ADDR_SHADOW Register .....	1405
10-30. SRC_BEG_ADDR_ACTIVE Register.....	1406
10-31. SRC_ADDR_ACTIVE Register.....	1407
10-32. DST_BEG_ADDR_SHADOW Register.....	1408
10-33. DST_ADDR_SHADOW Register .....	1409
10-34. DST_BEG_ADDR_ACTIVE Register .....	1410
10-35. DST_ADDR_ACTIVE Register .....	1411
11-1. ERAD Overview .....	1415
11-2. System Event Counter Inputs .....	1418
11-3. GLBL_EVENT_STAT Register .....	1424
11-4. GLBL_HALT_STAT Register .....	1426
11-5. GLBL_ENABLE Register .....	1428
11-6. GLBL_CTM_RESET Register .....	1430
11-7. GLBL_OWNER Register .....	1431
11-8. HWBP_MASK Register.....	1433
11-9. HWBP_REF Register .....	1434
11-10. HWBP_CLEAR Register .....	1435
11-11. HWBP_CNTL Register .....	1436
11-12. HWBP_STATUS Register.....	1438
11-13. CTM_CNTL Register.....	1440
11-14. CTM_STATUS Register .....	1442
11-15. CTM_REF Register .....	1443
11-16. CTM_COUNT Register .....	1444

11-17. CTM_MAX_COUNT Register.....	1445
11-18. CTM_INPUT_SEL Register.....	1446
11-19. CTM_CLEAR Register.....	1447
12-1. Analog Subsystem Block Diagram (100-Pin PZ LQFP).....	1450
12-2. Analog Subsystem Block Diagram (64-Pin PM LQFP).....	1451
12-3. Analog Subsystem Block Diagram (56-Pin RSH VQFN).....	1452
12-4. Analog Group Connections .....	1453
12-5. ANAREFPP Register .....	1461
12-6. TSNSCTL Register.....	1462
12-7. ANAREFCTL Register .....	1463
12-8. VMONCTL Register.....	1465
12-9. DCDCCTL Register .....	1466
12-10. DCDCSTS Register .....	1467
12-11. CMPHPMXSEL Register .....	1468
12-12. CMPLPMXSEL Register .....	1470
12-13. CMPHNMXSEL Register .....	1472
12-14. CMPLNMXSEL Register .....	1473
12-15. LOCK Register.....	1474
13-1. ADC Module Block Diagram .....	1478
13-2. SOC Block Diagram.....	1481
13-3. Single-Ended Input Model.....	1482
13-4. Round Robin Priority Example .....	1487
13-5. High Priority Example.....	1488
13-6. Burst Priority Example .....	1490
13-7. ADC EOC Interrupts .....	1491
13-8. ADC PPB Block Diagram .....	1493
13-9. ADC PPB Interrupt Event .....	1495
13-10. Opens/Shorts Detection Circuit.....	1496
13-11. Input Circuit Equivalent with OSDETECT Enabled .....	1497
13-12. ADC Timings for 12-bit Mode in Early Interrupt Mode .....	1500
13-13. ADC Timings for 12-bit Mode in Late Interrupt Mode .....	1501
13-14. Example: Basic Synchronous Operation .....	1502
13-15. Example: Synchronous Operation with Multiple Trigger Sources.....	1503
13-16. Example: Synchronous Operation with Uneven SOC Numbers .....	1504
13-17. Example: Asynchronous Operation with Uneven SOC Numbers – Trigger Overflow.....	1504
13-18. Example: Synchronous Equivalent Operation with Non-Overlapping Conversions .....	1505
13-19. ADC Shared Reference System.....	1507
13-20. ADC Shared Reference .....	1508
13-21. ADCCTL1 Register.....	1513
13-22. ADCCTL2 Register.....	1515
13-23. ADCBURSTCTL Register .....	1516
13-24. ADCINTFLG Register.....	1518
13-25. ADCINTFLGCLR Register .....	1520
13-26. ADCINTOVF Register .....	1522
13-27. ADCINTOVFCLR Register .....	1524
13-28. ADCINTSEL1N2 Register .....	1525
13-29. ADCINTSEL3N4 Register .....	1527
13-30. ADCSOCPRICL Register .....	1529
13-31. ADCINTSOCSEL1 Register .....	1532

13-32. ADCINTSOCSEL2 Register .....	1534
13-33. ADCSOCFLG1 Register.....	1536
13-34. ADCSOCFRC1 Register .....	1541
13-35. ADCSOCOVF1 Register .....	1546
13-36. ADCSOCOVFCLR1 Register.....	1550
13-37. ADCSOC0CTL Register.....	1554
13-38. ADCSOC1CTL Register.....	1556
13-39. ADCSOC2CTL Register.....	1558
13-40. ADCSOC3CTL Register.....	1560
13-41. ADCSOC4CTL Register.....	1562
13-42. ADCSOC5CTL Register.....	1564
13-43. ADCSOC6CTL Register.....	1566
13-44. ADCSOC7CTL Register.....	1568
13-45. ADCSOC8CTL Register.....	1570
13-46. ADCSOC9CTL Register.....	1572
13-47. ADCSOC10CTL Register .....	1574
13-48. ADCSOC11CTL Register .....	1576
13-49. ADCSOC12CTL Register .....	1578
13-50. ADCSOC13CTL Register .....	1580
13-51. ADCSOC14CTL Register .....	1582
13-52. ADCSOC15CTL Register .....	1584
13-53. ADCEVTSTAT Register .....	1586
13-54. ADCEVTCLR Register.....	1588
13-55. ADCEVTSEL Register.....	1590
13-56. ADCEVTINTSEL Register.....	1592
13-57. ADCOSDETECT Register.....	1594
13-58. ADCCOUNTER Register .....	1595
13-59. ADCREV Register.....	1596
13-60. ADCOFFTRIM Register .....	1597
13-61. ADCPPB1CONFIG Register.....	1598
13-62. ADCPPB1STAMP Register .....	1600
13-63. ADCPPB1OFFCAL Register.....	1601
13-64. ADCPPB1OFFREF Register.....	1602
13-65. ADCPPB1TRIPHI Register .....	1603
13-66. ADCPPB1TRIPL0 Register.....	1604
13-67. ADCPPB2CONFIG Register.....	1605
13-68. ADCPPB2STAMP Register .....	1607
13-69. ADCPPB2OFFCAL Register.....	1608
13-70. ADCPPB2OFFREF Register.....	1609
13-71. ADCPPB2TRIPHI Register .....	1610
13-72. ADCPPB2TRIPL0 Register.....	1611
13-73. ADCPPB3CONFIG Register.....	1612
13-74. ADCPPB3STAMP Register .....	1614
13-75. ADCPPB3OFFCAL Register.....	1615
13-76. ADCPPB3OFFREF Register.....	1616
13-77. ADCPPB3TRIPHI Register .....	1617
13-78. ADCPPB3TRIPL0 Register.....	1618
13-79. ADCPPB4CONFIG Register.....	1619
13-80. ADCPPB4STAMP Register .....	1621

13-81. ADCPPB4OFFCAL Register.....	1622
13-82. ADCPPB4OFFREF Register.....	1623
13-83. ADCPPB4TRIPHI Register .....	1624
13-84. ADCPPB4TRIPLO Register.....	1625
13-85. ADCINTCYCLE Register .....	1626
13-86. ADCINLTRIM1 Register .....	1627
13-87. ADCINLTRIM2 Register .....	1628
13-88. ADCINLTRIM3 Register .....	1629
13-89. ADCRESULT0 Register .....	1632
13-90. ADCRESULT1 Register .....	1633
13-91. ADCRESULT2 Register .....	1634
13-92. ADCRESULT3 Register .....	1635
13-93. ADCRESULT4 Register .....	1636
13-94. ADCRESULT5 Register .....	1637
13-95. ADCRESULT6 Register .....	1638
13-96. ADCRESULT7 Register .....	1639
13-97. ADCRESULT8 Register .....	1640
13-98. ADCRESULT9 Register .....	1641
13-99. ADCRESULT10 Register.....	1642
13-100. ADCRESULT11 Register .....	1643
13-101. ADCRESULT12 Register .....	1644
13-102. ADCRESULT13 Register .....	1645
13-103. ADCRESULT14 Register .....	1646
13-104. ADCRESULT15 Register .....	1647
13-105. ADCPPB1RESULT Register .....	1648
13-106. ADCPPB2RESULT Register .....	1649
13-107. ADCPPB3RESULT Register .....	1650
13-108. ADCPPB4RESULT Register .....	1651
14-1. PGA Block Diagram.....	1657
14-2. PGA Offset Trim .....	1659
14-3. PGA Gain Trim .....	1660
14-4. PGA_GND to Remote Ground .....	1660
14-5. PGA_GND to VSSA.....	1660
14-6. PGA Amplifier Example .....	1662
14-7. PGA Filter Example .....	1662
14-8. Buffered DAC Offset .....	1663
14-9. PGA Pin Alternate Functions .....	1664
14-10. PGACTL Register .....	1667
14-11. PGALOCK Register .....	1668
14-12. PGAGAIN3TRIM Register.....	1669
14-13. PGAGAIN6TRIM Register.....	1670
14-14. PGAGAIN12TRIM Register .....	1671
14-15. PGAGAIN24TRIM Register .....	1672
14-16. PGATYPE Register .....	1673
15-1. DAC Module Block Diagram .....	1675
15-2. DACREV Register .....	1680
15-3. DACCTL Register .....	1681
15-4. DACVALA Register .....	1682
15-5. DACVALS Register .....	1683

15-6.	DACOUTEN Register .....	1684
15-7.	DACLOCK Register .....	1685
15-8.	DACRIM Register .....	1686
16-1.	CMPSS Module Block Diagram .....	1690
16-2.	Comparator Block Diagram .....	1690
16-3.	Reference DAC Block Diagram .....	1691
16-4.	Output Voltage Calculation .....	1691
16-5.	Ramp Generator Block Diagram .....	1693
16-6.	Ramp Generator Behavior .....	1694
16-7.	Digital Filter Behavior .....	1694
16-8.	COMPCTL Register .....	1700
16-9.	COMPYSCTL Register .....	1702
16-10.	COMPSTS Register .....	1703
16-11.	COMPSTSCLR Register .....	1704
16-12.	COMPDACCTL Register .....	1705
16-13.	DACHVALS Register .....	1707
16-14.	DACHVALA Register .....	1708
16-15.	RAMPMAXREFA Register .....	1709
16-16.	RAMPMAXREFS Register .....	1710
16-17.	RAMPDECVALA Register .....	1711
16-18.	RAMPDECVALS Register .....	1712
16-19.	RAMPSTS Register .....	1713
16-20.	DACLVALS Register .....	1714
16-21.	DACLVALA Register .....	1715
16-22.	RAMPDLYA Register .....	1716
16-23.	RAMPDLYS Register .....	1717
16-24.	CTRIPLFILCTL Register .....	1718
16-25.	CTRIPLFILCLKCTL Register .....	1719
16-26.	CTRIPHFILCTL Register .....	1720
16-27.	CTRIPHFILCLKCTL Register .....	1721
16-28.	COMPLOCK Register .....	1722
17-1.	Sigma Delta Filter Module (SDFM) CPU Interface .....	1726
17-2.	Sigma Delta Filter Module (SDFM) Block Diagram .....	1728
17-3.	Block Diagram of One Filter Module .....	1729
17-4.	Different Modulator Modes Supported .....	1731
17-5.	Z-Transform of Sinc Filter of Order N .....	1732
17-6.	Simplified Sinc Filter Architecture .....	1732
17-7.	Frequency Response of different Sinc Filters .....	1733
17-8.	SDSYNC Event .....	1737
17-9.	Comparator Unit Structure .....	1739
17-10.	SDFM Error (SD_ERR) interrupt sources .....	1740
17-11.	SDFM Data Ready (SDy_DRINTx) Interrupt .....	1742
17-12.	SDIFLG Register .....	1746
17-13.	SDIFLGCLR Register .....	1749
17-14.	SDCTL Register .....	1751
17-15.	SDMFILEN Register .....	1752
17-16.	SDSTATUS Register .....	1753
17-17.	SDCTLPARM1 Register .....	1755
17-18.	SDDFPARM1 Register .....	1756

17-19. SDDPARAM1 Register .....	1757
17-20. SDCMPH1 Register .....	1758
17-21. SDCMPL1 Register .....	1759
17-22. SDCPARAM1 Register .....	1760
17-23. SDDATA1 Register .....	1762
17-24. SDDATFIFO1 Register .....	1763
17-25. SDCDATA1 Register .....	1764
17-26. SDCMPHZ1 Register .....	1765
17-27. SDFIFOCTL1 Register .....	1766
17-28. SDSYNC1 Register .....	1767
17-29. SDCTLPARAM2 Register .....	1768
17-30. SDDFPARM2 Register .....	1769
17-31. SDDPARAM2 Register .....	1770
17-32. SDCMPH2 Register .....	1771
17-33. SDCMPL2 Register .....	1772
17-34. SDCPARAM2 Register .....	1773
17-35. SDDATA2 Register .....	1775
17-36. SDDATFIFO2 Register .....	1776
17-37. SDCDATA2 Register .....	1777
17-38. SDCMPHZ2 Register .....	1778
17-39. SDFIFOCTL2 Register .....	1779
17-40. SDSYNC2 Register .....	1780
17-41. SDCTLPARAM3 Register .....	1781
17-42. SDDFPARM3 Register .....	1782
17-43. SDDPARAM3 Register .....	1783
17-44. SDCMPH3 Register .....	1784
17-45. SDCMPL3 Register .....	1785
17-46. SDCPARAM3 Register .....	1786
17-47. SDDATA3 Register .....	1788
17-48. SDDATFIFO3 Register .....	1789
17-49. SDCDATA3 Register .....	1790
17-50. SDCMPHZ3 Register .....	1791
17-51. SDFIFOCTL3 Register .....	1792
17-52. SDSYNC3 Register .....	1793
17-53. SDCTLPARAM4 Register .....	1794
17-54. SDDFPARM4 Register .....	1795
17-55. SDDPARAM4 Register .....	1796
17-56. SDCMPH4 Register .....	1797
17-57. SDCMPL4 Register .....	1798
17-58. SDCPARAM4 Register .....	1799
17-59. SDDATA4 Register .....	1801
17-60. SDDATFIFO4 Register .....	1802
17-61. SDCDATA4 Register .....	1803
17-62. SDCMPHZ4 Register .....	1804
17-63. SDFIFOCTL4 Register .....	1805
17-64. SDSYNC4 Register .....	1806
18-1. Multiple ePWM Modules .....	1814
18-2. Submodules and Signal Connections for an ePWM Module .....	1815
18-3. ePWM modules and Critical Internal Signal Interconnects .....	1817

18-4.	Time-Base Submodule .....	1820
18-5.	Time-Base Submodule Signals and Registers .....	1821
18-6.	Time-Base Frequency and Period .....	1823
18-7.	Time-Base Counter Synchronization Scheme.....	1825
18-8.	Time-Base Up-Count Mode Waveforms .....	1827
18-9.	Time-Base Down-Count Mode Waveforms .....	1828
18-10.	Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event ...	1828
18-11.	Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event .....	1829
18-12.	Global Load: Signals and Registers .....	1830
18-13.	Counter-Compare Submodule.....	1831
18-14.	Detailed View of the Counter-Compare Submodule.....	1832
18-15.	Counter-Compare Event Waveforms in Up-Count Mode .....	1835
18-16.	Counter-Compare Events in Down-Count Mode .....	1836
18-17.	Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event .....	1837
18-18.	Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event .....	1837
18-19.	Action-Qualifier Submodule .....	1838
18-20.	Action-Qualifier Submodule Inputs and Outputs .....	1839
18-21.	Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs .....	1840
18-22.	AQCTL[SHDWAQAMODE] .....	1843
18-23.	AQCTL[SHDWAQBMODE] .....	1843
18-24.	Up-Down-Count Mode Symmetrical Waveform .....	1845
18-25.	Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High .....	1846
18-26.	Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low .....	1847
18-27.	Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA .....	1848
18-28.	Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low .....	1848
18-29.	Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary .....	1849
18-30.	Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low .....	1850
18-31.	Up-Down-Count, PWM Waveform Generation Utilizing T1 and T2 Events .....	1850
18-32.	Dead_Band Submodule .....	1851
18-33.	Configuration Options for the Dead-Band Submodule .....	1853
18-34.	Dead-Band Waveforms for Typical Cases (0% < Duty < 100%).....	1855
18-35.	PWM Chopper Submodule.....	1857
18-36.	PWM Chopper Submodule Operational Details.....	1858
18-37.	Simple PWM Chopper Submodule Waveforms Showing Chopping Action Only .....	1858
18-38.	PWM Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses .....	1859
18-39.	PWM Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses.....	1860
18-40.	Trip-Zone Submodule.....	1861
18-41.	Trip-Zone Submodule Mode Control Logic .....	1865
18-42.	Trip-Zone Submodule Interrupt Logic.....	1866
18-43.	Event-Trigger Submodule .....	1867
18-44.	Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs.....	1868
18-45.	Event-Trigger Interrupt Generator.....	1870
18-46.	Event-Trigger SOCA Pulse Generator .....	1871



18-47. Event-Trigger SOCB Pulse Generator .....	1871
18-48. Digital-Compare Submodule High-Level Block Diagram .....	1872
18-49. GPIO MUX-to-Trip Input Connectivity .....	1873
18-50. DCAEVT1 Event Triggering.....	1876
18-51. DCAEVT2 Event Triggering.....	1876
18-52. DCBEVT1 Event Triggering.....	1877
18-53. DCBEVT2 Event Triggering.....	1877
18-54. Event Filtering .....	1878
18-55. Blanking Window Timing Diagram .....	1879
18-56. Valley Switching .....	1880
18-57. ePWM X-BAR .....	1881
18-58. ePWM XBAR Architecture - Single Output .....	1882
18-59. Simplified ePWM Module.....	1884
18-60. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave .....	1885
18-61. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$ .....	1886
18-62. Buck Waveforms for (Note: Only three bucks shown here) .....	1887
18-63. Control of Four Buck Stages. (Note: $F_{PWM2} = N \times F_{PWM1}$ ).....	1888
18-64. Buck Waveforms for (Note: $F_{PWM2} = F_{PWM1}$ ).....	1889
18-65. Control of Two Half-H Bridge Stages ( $F_{PWM2} = N \times F_{PWM1}$ ).....	1890
18-66. Half-H Bridge Waveforms for (Note: Here $F_{PWM2} = F_{PWM1}$ ).....	1891
18-67. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control .....	1892
18-68. 3-Phase Inverter Waveforms for (Only One Inverter Shown) .....	1893
18-69. Configuring Two PWM Modules for Phase Control.....	1894
18-70. Timing Waveforms Associated With Phase Control Between Two Modules .....	1895
18-71. Control of a 3-Phase Interleaved DC/DC Converter .....	1896
18-72. 3-Phase Interleaved DC/DC Converter Waveforms for .....	1897
18-73. Controlling a Full-H Bridge Stage ( $F_{PWM2} = F_{PWM1}$ ) .....	1898
18-74. ZVS Full-H Bridge Waveforms .....	1899
18-75. Peak Current Mode Control of a Buck Converter .....	1900
18-76. Peak Current Mode Control Waveforms for .....	1900
18-77. Control of Two Resonant Converter Stages .....	1901
18-78. H-Bridge LLC Resonant Converter PWM Waveforms.....	1901
18-79. Resolution Calculations for Conventionally Generated PWM .....	1903
18-80. Operating Logic Using MEP .....	1905
18-81. HRPWM Extension Registers and Memory Configuration.....	1906
18-82. HRPWM System Interface .....	1907
18-83. HRPWM Block Diagram .....	1908
18-84. HRPWM and HRCAL Source Clock .....	1908
18-85. Required PWM Waveform for a Requested Duty = 40.5% .....	1911
18-86. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0) .....	1914
18-87. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0) .....	1916
18-88. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1) .....	1916
18-89. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1).....	1917
18-90. Simple Buck Controlled Converter Using a Single PWM .....	1922
18-91. PWM Waveform Generated for Simple Buck Controlled Converter .....	1922
18-92. Simple Reconstruction Filter for a PWM-based DAC .....	1924
18-93. PWM Waveform Generated for the PWM DAC Function .....	1924
18-94. TBCTL Register .....	1934
18-95. TBCTL2 Register .....	1937



18-96. TBCTR Register .....	1938
18-97. TBSTS Register .....	1939
18-98. CMPCTL Register .....	1940
18-99. CMPCTL2 Register .....	1942
18-100. DBCTL Register .....	1944
18-101. DBCTL2 Register .....	1947
18-102. AQCTL Register .....	1948
18-103. AQTSRCSEL Register .....	1950
18-104. PCCTL Register .....	1951
18-105. VCAPCTL Register .....	1953
18-106. VCNTCFG Register .....	1955
18-107. HRCNFG Register .....	1957
18-108. HRPWR Register .....	1959
18-109. HRMSTEP Register .....	1960
18-110. HRCNFG2 Register .....	1961
18-111. HRPCTL Register .....	1962
18-112. TRREM Register .....	1964
18-113. GLDCTL Register .....	1965
18-114. GLDCFG Register .....	1967
18-115. EPWMXLINK Register .....	1969
18-116. AQCTLA Register .....	1973
18-117. AQCTLA2 Register .....	1975
18-118. AQCTLB Register .....	1977
18-119. AQCTLB2 Register .....	1979
18-120. AQSFRC Register .....	1981
18-121. AQCSFRC Register .....	1983
18-122. DBREDHR Register .....	1984
18-123. DBRED Register .....	1985
18-124. DBFEDHR Register .....	1986
18-125. DBFED Register .....	1987
18-126. TBPFS Register .....	1988
18-127. TBPRDHR Register .....	1989
18-128. TBPRD Register .....	1990
18-129. CMPA Register .....	1991
18-130. CMPB Register .....	1992
18-131. CMPC Register .....	1993
18-132. CMPD Register .....	1994
18-133. GLDCTL2 Register .....	1995
18-134. SWVDELVAL Register .....	1996
18-135. TZSEL Register .....	1997
18-136. TZDCSEL Register .....	1999
18-137. TZCTL Register .....	2001
18-138. TZCTL2 Register .....	2003
18-139. TZCTLDCA Register .....	2005
18-140. TZCTLDCB Register .....	2007
18-141. TZEINT Register .....	2009
18-142. TZFLG Register .....	2010
18-143. TZCBCFLG Register .....	2012
18-144. TZOSTFLG Register .....	2014

18-145. TZCLR Register .....	2016
18-146. TZCBCCLR Register .....	2018
18-147. TZOSTCLR Register .....	2020
18-148. TZFRC Register.....	2022
18-149. ETSEL Register .....	2023
18-150. ETPS Register .....	2026
18-151. ETFLG Register .....	2029
18-152. ETCLR Register.....	2030
18-153. ETFRC Register.....	2031
18-154. ETINTPS Register .....	2032
18-155. ETSOCPS Register.....	2033
18-156. ETCNTINITCTL Register .....	2035
18-157. ETCNTINIT Register .....	2036
18-158. DCTRIPESEL Register .....	2037
18-159. DCACTL Register.....	2039
18-160. DCBCTL Register.....	2041
18-161. DCFCTL Register .....	2043
18-162. DCCAPCTL Register .....	2045
18-163. DCFOFFSET Register .....	2047
18-164. DCFOFFSETCNT Register.....	2048
18-165. DCFWINDOW Register .....	2049
18-166. DCFWINDOWCNT Register.....	2050
18-167. DCCAP Register .....	2051
18-168. DCAHTRIPSEL Register.....	2052
18-169. DCALTRIPSEL Register .....	2054
18-170. DCBHTRIPSEL Register.....	2056
18-171. DCBLTRIPSEL Register .....	2058
18-172. EPWMLOCK Register.....	2060
18-173. HWVDELVAL Register .....	2062
18-174. VCNTVAL Register .....	2063
18-175. SYNCSELECT Register.....	2065
18-176. ADCSOCOUTSELECT Register.....	2068
18-177. SYNCSOCLOCK Register .....	2071
19-1. Capture and APWM Modes of Operation .....	2084
19-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode.....	2085
19-3. eCAP Block Diagram .....	2086
19-4. Event Prescale Control .....	2087
19-5. Prescale Function Waveforms.....	2087
19-6. Details of the Continuous/One-shot Block .....	2088
19-7. Details of the Counter and Synchronization Block.....	2089
19-8. Time-Base Counter Synchronization Scheme .....	2090
19-9. Interrupts in eCAP Module .....	2092
19-10. PWM Waveform Details Of APWM Mode Operation .....	2093
19-11. Time-Base Frequency and Period Calculation .....	2094
19-12. Capture Sequence for Absolute Time-stamp and Rising Edge Detect.....	2095
19-13. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect.....	2096
19-14. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect .....	2097
19-15. Capture Sequence for Delta Mode Time-stamp With Rising and Falling Edge Detect .....	2098
19-16. PWM Waveform Details of APWM Mode Operation.....	2099

19-17. TSCTR Register .....	2102
19-18. CTRPHS Register .....	2103
19-19. CAP1 Register .....	2104
19-20. CAP2 Register .....	2105
19-21. CAP3 Register .....	2106
19-22. CAP4 Register .....	2107
19-23. ECCTL0 Register .....	2108
19-24. ECCTL1 Register .....	2109
19-25. ECCTL2 Register .....	2111
19-26. ECEINT Register .....	2114
19-27. ECFLG Register .....	2116
19-28. ECCLR Register .....	2118
19-29. ECFRC Register .....	2119
20-1. HRCAP Operations Block Diagram .....	2124
20-2. HRCAP Calibration .....	2126
20-3. HRCTL Register .....	2130
20-4. HRINTEN Register .....	2132
20-5. HRFLG Register .....	2133
20-6. HRCLR Register .....	2134
20-7. HRFRC Register .....	2135
20-8. HRCALPRD Register .....	2136
20-9. HRSYSCLKCTR Register .....	2137
20-10. HRSYSCLKCAP Register .....	2138
20-11. HRCLKCTR Register .....	2139
20-12. HRCLKCAP Register .....	2140
21-1. Optical Encoder Disk .....	2142
21-2. QEP Encoder Output Signal for Forward/Reverse Movement .....	2142
21-3. Index Pulse Example .....	2143
21-4. Functional Block Diagram of the eQEP Peripheral .....	2145
21-5. Functional Block Diagram of Decoder Unit .....	2147
21-6. Quadrature Decoder State Machine .....	2148
21-7. Quadrature-clock and Direction Decoding .....	2149
21-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or 0xF9F) .....	2151
21-9. Position Counter Underflow/Overflow (QPOSMAX = 4) .....	2152
21-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1) .....	2153
21-11. Strobe Event Latch (QEPCTL[SEL] = 1) .....	2154
21-12. eQEP Position-compare Unit .....	2155
21-13. eQEP Position-compare Event Generation Points .....	2156
21-14. eQEP Position-compare Sync Output Pulse Stretcher .....	2156
21-15. eQEP Edge Capture Unit .....	2158
21-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010) .....	2158
21-17. eQEP Edge Capture Unit - Timing Details .....	2159
21-18. eQEP Watchdog Timer .....	2160
21-19. eQEP Unit Time Base .....	2161
21-20. QMA Module Block Diagram .....	2162
21-21. QMA Mode-1 .....	2163
21-22. QMA Mode-2 .....	2164
21-23. QMA Mode-3 .....	2165
21-24. EQEP Interrupt Generation .....	2166

21-25. QPOSCNT Register .....	2170
21-26. QPOSINIT Register .....	2171
21-27. QPOSMAX Register .....	2172
21-28. QPOSCMP Register .....	2173
21-29. QPOSILAT Register .....	2174
21-30. QPOSSLAT Register .....	2175
21-31. QPOSLAT Register .....	2176
21-32. QUTMR Register .....	2177
21-33. QUPRD Register .....	2178
21-34. QWDTMR Register .....	2179
21-35. QWDPRD Register .....	2180
21-36. QDECCTL Register .....	2181
21-37. QEPCTL Register .....	2183
21-38. QCAPCTL Register .....	2186
21-39. QPOSCTL Register .....	2187
21-40. QEINT Register .....	2188
21-41. QFLG Register .....	2190
21-42. QCLR Register .....	2192
21-43. QFRC Register .....	2194
21-44. QEPSTS Register .....	2196
21-45. QCTMR Register .....	2198
21-46. QCPRD Register .....	2199
21-47. QCTMRLAT Register .....	2200
21-48. QCPRDLAT Register .....	2201
21-49. REV Register .....	2202
21-50. QEPSTROBESEL Register .....	2203
21-51. QMACTRL Register .....	2204
22-1. SPI CPU Interface .....	2209
22-2. SPI Interrupt Flags and Enable Logic Generation .....	2211
22-3. SPI DMA Trigger Diagram .....	2212
22-4. SPI Master/Slave Connection .....	2213
22-5. SPI Module Master Configuration .....	2214
22-6. SPI Module Slave Configuration .....	2215
22-7. SPICLK Signal Options .....	2218
22-8. SPI: SPICLK-LSPCLK Characteristic When (BRR + 1) is Odd, BRR > 3, and CLKPOLARITY = 1 .....	2218
22-9. SPI 3-wire Master Mode .....	2221
22-10. SPI 3-wire Slave Mode .....	2221
22-11. Five Bits per Character .....	2224
22-12. SPI Digital Audio Receiver Configuration Using Two SPIs .....	2226
22-13. Standard Right-Justified Digital Audio Data Format .....	2226
22-14. SPICCR Register .....	2229
22-15. SPICTL Register .....	2231
22-16. SPISTS Register .....	2233
22-17. SPIBRR Register .....	2235
22-18. SPIRXEMU Register .....	2236
22-19. SPIRXBUF Register .....	2237
22-20. SPITXBUF Register .....	2238
22-21. SPIDAT Register .....	2239
22-22. SPIFFTX Register .....	2240

22-23. SPIFFRX Register.....	2242
22-24. SPIFFCT Register.....	2244
22-25. SPIPRI Register .....	2245
23-1. SCI CPU Interface .....	2250
23-2. Serial Communications Interface (SCI) Module Block Diagram .....	2251
23-3. Typical SCI Data Frame Formats .....	2253
23-4. Idle-Line Multiprocessor Communication Format .....	2255
23-5. Double-Buffered WUT and TXSHF .....	2256
23-6. Address-Bit Multiprocessor Communication Format.....	2257
23-7. SCI Asynchronous Communications Format .....	2258
23-8. SCI RX Signals in Communication Modes.....	2258
23-9. SCI TX Signals in Communications Mode .....	2259
23-10. SCI FIFO Interrupt Flags and Enable Logic .....	2261
23-11. SCICCR Register.....	2265
23-12. SCICTL1 Register .....	2267
23-13. SCIHBAUD Register .....	2269
23-14. SCILBAUD Register .....	2270
23-15. SCICTL2 Register .....	2271
23-16. SCIRXST Register .....	2273
23-17. SCIRXEMU Register.....	2275
23-18. SCIRXBUF Register .....	2276
23-19. SCITXBUF Register.....	2277
23-20. SCIFFTX Register.....	2278
23-21. SCIFFRX Register .....	2280
23-22. SCIFFCT Register.....	2282
23-23. SCIPRI Register .....	2283
24-1. Multiple I2C Modules Connected.....	2287
24-2. I2C Module Conceptual Block Diagram .....	2289
24-3. Clocking Diagram for the I2C Module .....	2289
24-4. The Roles of the Clock Divide-Down Values (ICCL and ICCH).....	2290
24-5. Bit Transfer on the I2C bus .....	2291
24-6. I2C Module START and STOP Conditions .....	2293
24-7. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown) .....	2294
24-8. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR) .....	2294
24-9. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR) .....	2294
24-10. I2C Module Free Data Format (FDF = 1 in I2CMDR) .....	2295
24-11. Repeated START Condition (in This Case, 7-Bit Addressing Format).....	2295
24-12. Synchronization of Two I2C Clock Generators During Arbitration .....	2296
24-13. Arbitration Procedure Between Two Master-Transmitters .....	2297
24-14. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit .....	2298
24-15. Enable Paths of the I2C Interrupt Requests .....	2299
24-16. Backwards Compatibility Mode Bit, Slave Transmitter .....	2300
24-17. I2C_FIFO_interrupt.....	2301
24-18. I2COAR Register .....	2304
24-19. I2CIER Register .....	2305
24-20. I2CSTR Register .....	2306
24-21. I2CCLKL Register .....	2310
24-22. I2CCLKH Register.....	2311
24-23. I2CCNT Register .....	2312

24-24. I2CDRR Register .....	2313
24-25. I2CSAR Register .....	2314
24-26. I2CDXR Register .....	2315
24-27. I2CMDR Register.....	2316
24-28. I2CISRC Register .....	2320
24-29. I2CEMDR Register .....	2321
24-30. I2CPSC Register .....	2322
24-31. I2CFFTX Register .....	2323
24-32. I2CFFRX Register.....	2325
25-1. PMBus Module Conceptual Block Diagram.....	2331
25-2. Quick Command Message .....	2332
25-3. Send Byte Message with and without PEC .....	2333
25-4. Receive Byte Message with and without PEC .....	2333
25-5. Write Byte and Write Word Messages with and without PEC .....	2334
25-6. Read Byte and Read Word Messages with and without PEC.....	2335
25-7. Process Call Message with and without PEC .....	2336
25-8. Block Write Message with and without PEC.....	2336
25-9. Block Read Message with and without PEC.....	2337
25-10. Block Write-Block Read Process Call Message with and without PEC .....	2338
25-11. Alert Response Message.....	2338
25-12. Extended Command Write Byte and Write Word Messages with and without PEC .....	2339
25-13. Extended Command Write Byte and Write Word Messages with and without PEC .....	2339
25-14. Group Command Message with and without PEC.....	2340
25-15. Quick Command Message .....	2341
25-16. Send Byte Message with and without PEC .....	2341
25-17. Receive Byte Message with and without PEC .....	2342
25-18. Write Byte and Write Word Messages with and without PEC .....	2342
25-19. Read Byte and Read Word Messages with and without PEC.....	2343
25-20. Process Call Message with and without PEC .....	2344
25-21. Block Write Message with and without PEC.....	2344
25-22. Block Read Message with and without PEC.....	2345
25-23. Block Write-Block Read Process Call Message with and without PEC .....	2346
25-24. Alert Response Message.....	2346
25-25. Extended Command Write Message with and without PEC.....	2347
25-26. Extended Command Read Message with and without PEC .....	2347
25-27. Group Command Message with and without PEC.....	2348
25-28. PMBMC Register .....	2351
25-29. PMBTXBUF Register .....	2353
25-30. PMBRXBUF Register .....	2354
25-31. PMBACK Register.....	2355
25-32. PMBSTS Register .....	2356
25-33. PMBINTM Register .....	2358
25-34. PMBSC Register .....	2360
25-35. PMBHSA Register.....	2362
25-36. PMBCTRL Register .....	2363
25-37. PMBTIMCTL Register .....	2366
25-38. PMBTIMCLK Register .....	2367
25-39. PMBTIMSTSETUP Register .....	2368
25-40. PMBTIMBIDLE Register.....	2369

25-41. PMBTIMLOWTIMOUT Register .....	2370
25-42. PMBTIMHIGHTIMOUT Register .....	2371
26-1. CAN Block Diagram.....	2374
26-2. CAN_MUX.....	2380
26-3. CAN Core in Silent Mode .....	2380
26-4. CAN Core in Loopback Mode .....	2381
26-5. CAN Core in External Loopback Mode.....	2382
26-6. CAN Core in Loopback Combined with Silent Mode .....	2382
26-7. CAN Interrupt Topology 1 .....	2384
26-8. CAN Interrupt Topology 2 .....	2385
26-9. Initialization of a Transmit Object .....	2387
26-10. Initialization of a Single Receive Object for Data Frames .....	2387
26-11. Initialization of a single Receive Object for Remote Frames .....	2388
26-12. CPU Handling of a FIFO Buffer (Interrupt Driven) .....	2393
26-13. Bit Timing.....	2394
26-14. The Propagation Time Segment .....	2395
26-15. Synchronization on Late and Early Edges .....	2397
26-16. Filtering of Short Dominant Spikes.....	2398
26-17. Structure of the CAN Core's CAN Protocol Controller .....	2399
26-18. Data Transfer Between IF1 / IF2 Registers and Message RAM .....	2403
26-19. Structure of a Message Object .....	2404
26-20. Message RAM Representation in Debug Mode.....	2407
26-21. CAN_CTL Register.....	2411
26-22. CAN_ES Register .....	2414
26-23. CAN_ERRC Register .....	2416
26-24. CAN_BTR Register .....	2417
26-25. CAN_INT Register .....	2419
26-26. CAN_TEST Register.....	2420
26-27. CAN_PERR Register .....	2422
26-28. CAN_RAM_INIT Register .....	2423
26-29. CAN_GLB_INT_EN Register .....	2424
26-30. CAN_GLB_INT_FLG Register.....	2425
26-31. CAN_GLB_INT_CLR Register.....	2426
26-32. CAN_ABOTR Register .....	2427
26-33. CAN_TXRQ_X Register .....	2428
26-34. CAN_TXRQ_21 Register .....	2429
26-35. CAN_NDAT_X Register .....	2430
26-36. CAN_NDAT_21 Register .....	2431
26-37. CAN_IPEN_X Register .....	2432
26-38. CAN_IPEN_21 Register .....	2433
26-39. CAN_MVAL_X Register .....	2434
26-40. CAN_MVAL_21 Register .....	2435
26-41. CAN_IP_MUX21 Register.....	2436
26-42. CAN_IF1CMD Register.....	2437
26-43. CAN_IF1MSK Register .....	2441
26-44. CAN_IF1ARB Register .....	2442
26-45. CAN_IF1MCTL Register .....	2444
26-46. CAN_IF1DATA Register.....	2447
26-47. CAN_IF1DATB Register.....	2448



26-48. CAN_IF2CMD Register.....	2449
26-49. CAN_IF2MSK Register.....	2453
26-50. CAN_IF2ARB Register.....	2454
26-51. CAN_IF2MCTL Register.....	2456
26-52. CAN_IF2DATA Register.....	2459
26-53. CAN_IF2DATB Register.....	2460
26-54. CAN_IF3OBS Register.....	2461
26-55. CAN_IF3MSK Register.....	2463
26-56. CAN_IF3ARB Register.....	2464
26-57. CAN_IF3MCTL Register.....	2465
26-58. CAN_IF3DATA Register.....	2467
26-59. CAN_IF3DATB Register.....	2468
26-60. CAN_IF3UPD Register.....	2469
27-1. SCI Block Diagram.....	2476
27-2. SCI/LIN Block Diagram.....	2477
27-3. Typical SCI Data Frame Formats.....	2478
27-4. Asynchronous Communication Bit Timing.....	2479
27-5. Superfractional Divider Example.....	2482
27-6. Idle-Line Multiprocessor Communication Format.....	2484
27-7. Address-Bit Multiprocessor Communication Format.....	2484
27-8. Receive Buffers.....	2485
27-9. Transmit Buffers.....	2486
27-10. General Interrupt Scheme.....	2487
27-11. Interrupt Generation for Given Flags.....	2488
27-12. LIN Protocol Message Frame Format: Master Header and Slave Response.....	2496
27-13. Header 3 Fields: Sync Break, Sync, and ID.....	2496
27-14. Response Format of LIN Message Frame.....	2497
27-15. Message Header in Terms of $T_{bit}$ .....	2500
27-16. ID Field.....	2500
27-17. Measurements for Synchronization.....	2502
27-18. Synchronization Validation Process and Baud Rate Adjustment.....	2503
27-19. Optional Embedded Checksum in Response for Extended Frames.....	2504
27-20. Checksum Compare and Send for Extended Frames.....	2505
27-21. TXRX Error Detector.....	2507
27-22. Classic Checksum Generation at Transmitting Node.....	2508
27-23. LIN 2.0-Compliant Checksum Generation at Transmitting Node.....	2508
27-24. ID Reception, Filtering and Validation.....	2510
27-25. LIN Message Frame Showing LIN Interrupt Timing and Sequence.....	2512
27-26. Wakeup Signal Generation.....	2517
27-27. SCIGCR0 Register.....	2522
27-28. SCIGCR1 Register.....	2523
27-29. SCIGCR2 Register.....	2528
27-30. SCISSETINT Register.....	2530
27-31. SCICLEARINT Register.....	2534
27-32. SCISSETINTLVL Register.....	2538
27-33. SCICLEARINTLVL Register.....	2541
27-34. SCIFLR Register.....	2545
27-35. SCIINTVECT0 Register.....	2554
27-36. SCIINTVECT1 Register.....	2555

27-37. SCIFORMAT Register .....	2556
27-38. BRSR Register .....	2558
27-39. SCIED Register.....	2560
27-40. SCIRD Register.....	2561
27-41. SCITD Register.....	2562
27-42. SCIPIO0 Register .....	2563
27-43. SCIPIO2 Register .....	2564
27-44. LINCOMP Register.....	2565
27-45. LINRD0 Register .....	2566
27-46. LINRD1 Register .....	2567
27-47. LINMASK Register .....	2568
27-48. LINID Register .....	2569
27-49. LINTD0 Register.....	2570
27-50. LINTD1 Register.....	2571
27-51. MBRSR Register .....	2572
27-52. IODFTCTRL Register.....	2573
27-53. LIN_GLB_INT_EN Register.....	2576
27-54. LIN_GLB_INT_FLG Register .....	2577
27-55. LIN_GLB_INT_CLR Register .....	2578
28-1. FSI Transmitter (FSITX) CPU Interface .....	2581
28-2. FSI Receiver (FSIRX) CPU Interface.....	2581
28-3. FSI Receiver (FSIRX) CPU Interface with CLB .....	2582
28-4. FSI Transmitter Block Diagram.....	2587
28-5. FSI Transmitter Core Block Diagram .....	2588
28-6. FSI Receiver Block Diagram.....	2593
28-7. FSI Receiver Core Block Diagram .....	2593
28-8. Delay Line Control Circuit .....	2596
28-9. Flush Sequence Signals.....	2602
28-10. FSI with Internal Loopback.....	2602
28-11. FSITX as SPI Master, Transmit Only .....	2604
28-12. FSIRX as SPI Slave, Receive Only .....	2605
28-13. FSITX and FSIRX as SPI Master, Full Duplex .....	2607
28-14. Point to Point Connection .....	2608
28-15. RX_MASTER_CTRL Register.....	2613
28-16. RX_OPER_CTRL Register .....	2615
28-17. RX_FRAME_INFO Register .....	2617
28-18. RX_FRAME_TAG_UDATA Register.....	2618
28-19. RX_DMA_CTRL Register .....	2619
28-20. RX_EVT_STS Register.....	2620
28-21. RX_CRC_INFO Register .....	2623
28-22. RX_EVT_CLR Register.....	2624
28-23. RX_EVT_FRC Register .....	2626
28-24. RX_BUF_PTR_LOAD Register.....	2629
28-25. RX_BUF_PTR_STS Register.....	2630
28-26. RX_FRAME_WD_CTRL Register.....	2631
28-27. RX_FRAME_WD_REF Register .....	2632
28-28. RX_FRAME_WD_CNT Register .....	2633
28-29. RX_PING_WD_CTRL Register.....	2634
28-30. RX_PING_TAG Register .....	2635

28-31. RX_PING_WD_REF Register .....	2636
28-32. RX_PING_WD_CNT Register .....	2637
28-33. RX_INT1_CTRL Register .....	2638
28-34. RX_INT2_CTRL Register .....	2641
28-35. RX_LOCK_CTRL Register .....	2644
28-36. RX_ECC_DATA Register .....	2645
28-37. RX_ECC_VAL Register .....	2646
28-38. RX_ECC_SEC_DATA Register .....	2647
28-39. RX_ECC_LOG Register .....	2648
28-40. RX_DLYLINE_CTRL Register .....	2649
28-41. RX_VIS_1 Register .....	2650
28-42. RX_BUF_BASE_y Register .....	2651
28-43. TX_MASTER_CTRL Register .....	2654
28-44. TX_CLK_CTRL Register .....	2655
28-45. TX_OPER_CTRL_LO Register .....	2656
28-46. TX_OPER_CTRL_HI Register .....	2658
28-47. TX_FRAME_CTRL Register .....	2659
28-48. TX_FRAME_TAG_UDATA Register .....	2661
28-49. TX_BUF_PTR_LOAD Register .....	2662
28-50. TX_BUF_PTR_STS Register .....	2663
28-51. TX_PING_CTRL Register .....	2664
28-52. TX_PING_TAG Register .....	2665
28-53. TX_PING_TO_REF Register .....	2666
28-54. TX_PING_TO_CNT Register .....	2667
28-55. TX_INT_CTRL Register .....	2668
28-56. TX_DMA_CTRL Register .....	2670
28-57. TX_LOCK_CTRL Register .....	2671
28-58. TX_EVT_STS Register .....	2672
28-59. TX_EVT_CLR Register .....	2674
28-60. TX_EVT_FRC Register .....	2675
28-61. TX_USER_CRC Register .....	2676
28-62. TX_ECC_DATA Register .....	2677
28-63. TX_ECC_VAL Register .....	2678
28-64. TX_BUF_BASE_y Register .....	2679
29-1. Block Diagram of the CLB Subsystem in the Device .....	2685
29-2. Block Diagram of a CLB Tile and CPU Interface .....	2686
29-3. CLB Input Mux and Filter .....	2687
29-4. GPIO to CLB Tile Connections .....	2690
29-5. CLB Outputs .....	2691
29-6. Peripheral Signal Multiplexer .....	2693
29-7. The CLB Tile Submodules .....	2694
29-8. Counter Block .....	2696
29-9. FSM Block .....	2698
29-10. FSM LUT Block .....	2699
29-11. LUT4 Block .....	2699
29-12. Output LUT Block .....	2700
29-13. High Level Controller Block .....	2700
29-14. ....	2700
29-15. CLB_COUNT_RESET Register .....	2708

29-16. CLB_COUNT_MODE_1 Register .....	2709
29-17. CLB_COUNT_MODE_0 Register .....	2710
29-18. CLB_COUNT_EVENT Register .....	2711
29-19. CLB_FSM_EXTRA_IN0 Register .....	2712
29-20. CLB_FSM_EXTERNAL_IN0 Register .....	2713
29-21. CLB_FSM_EXTERNAL_IN1 Register .....	2714
29-22. CLB_FSM_EXTRA_IN1 Register .....	2715
29-23. CLB_LUT4_IN0 Register .....	2716
29-24. CLB_LUT4_IN1 Register .....	2717
29-25. CLB_LUT4_IN2 Register .....	2718
29-26. CLB_LUT4_IN3 Register .....	2719
29-27. CLB_FSM_LUT_FN1_0 Register .....	2720
29-28. CLB_FSM_LUT_FN2 Register .....	2721
29-29. CLB_LUT4_FN1_0 Register .....	2722
29-30. CLB_LUT4_FN2 Register .....	2723
29-31. CLB_FSM_NEXT_STATE_0 Register.....	2724
29-32. CLB_FSM_NEXT_STATE_1 Register.....	2725
29-33. CLB_FSM_NEXT_STATE_2 Register.....	2726
29-34. CLB_MISC_CONTROL Register.....	2727
29-35. CLB_OUTPUT_LUT_0 Register .....	2730
29-36. CLB_OUTPUT_LUT_1 Register .....	2731
29-37. CLB_OUTPUT_LUT_2 Register .....	2732
29-38. CLB_OUTPUT_LUT_3 Register .....	2733
29-39. CLB_OUTPUT_LUT_4 Register .....	2734
29-40. CLB_OUTPUT_LUT_5 Register .....	2735
29-41. CLB_OUTPUT_LUT_6 Register .....	2736
29-42. CLB_OUTPUT_LUT_7 Register .....	2737
29-43. CLB_HLC_EVENT_SEL Register.....	2738
29-44. CLB_LOAD_EN Register.....	2741
29-45. CLB_LOAD_ADDR Register.....	2742
29-46. CLB_LOAD_DATA Register .....	2743
29-47. CLB_INPUT_FILTER Register .....	2744
29-48. CLB_IN_MUX_SEL_0 Register .....	2746
29-49. CLB_LCL_MUX_SEL_1 Register .....	2748
29-50. CLB_LCL_MUX_SEL_2 Register .....	2749
29-51. CLB_BUF_PTR Register .....	2750
29-52. CLB_GP_REG Register .....	2751
29-53. CLB_OUT_EN Register .....	2752
29-54. CLB_GLBL_MUX_SEL_1 Register .....	2753
29-55. CLB_GLBL_MUX_SEL_2 Register .....	2754
29-56. CLB_INTR_TAG_REG Register .....	2755
29-57. CLB_LOCK Register .....	2756
29-58. CLB_DBG_R0 Register .....	2757
29-59. CLB_DBG_R1 Register .....	2758
29-60. CLB_DBG_R2 Register .....	2759
29-61. CLB_DBG_R3 Register .....	2760
29-62. CLB_DBG_C0 Register .....	2761
29-63. CLB_DBG_C1 Register .....	2762
29-64. CLB_DBG_C2 Register .....	2763

---

29-65. CLB_DBG_OUT Register .....	<a href="#">2764</a>
29-66. CLB_PUSH_y Register .....	<a href="#">2767</a>
29-67. CLB_PULL_y Register .....	<a href="#">2768</a>

## List of Tables

1-1.	C2000Ware Root Directories .....	82
2-1.	TMU Supported Instructions .....	85
2-2.	Viterbi Decode Performance .....	86
2-3.	Complex Math Performance .....	86
3-1.	Access to EALLOW-Protected Registers .....	88
3-2.	Reset Signals .....	89
3-3.	PIE Channel Mapping .....	95
3-4.	CPU Interrupt Vectors .....	96
3-5.	PIE Interrupt Vectors.....	97
3-6.	Clock Connections Sorted by Clock Domain.....	108
3-7.	Clock Connections Sorted by Module Name.....	110
3-8.	Example Watchdog Key Sequences .....	116
3-9.	Software Emulated STANDBY vs STANDBY Mode .....	118
3-10.	Local Shared RAM.....	122
3-11.	Global Shared RAM .....	122
3-12.	Error Handling in Different Scenarios .....	126
3-13.	Mapping of ECC Bits in Read Data from ECC/Parity Address Map .....	127
3-14.	Mapping of Parity Bits in Read Data from ECC/Parity Address Map .....	127
3-15.	RAM/Flash Status .....	139
3-16.	Security Levels .....	140
3-17.	Default Value of ZxOTP_CSMPWD1 and other fields (programmed by TI) .....	140
3-18.	System Control Base Address Table.....	155
3-19.	ACCESS_PROTECTION_REGS Registers.....	156
3-20.	ACCESS_PROTECTION_REGS Access Type Codes .....	156
3-21.	NMAVFLG Register Field Descriptions .....	158
3-22.	NMAVSET Register Field Descriptions .....	160
3-23.	NMAVCLR Register Field Descriptions .....	162
3-24.	NMAVINTEN Register Field Descriptions .....	164
3-25.	NMCPURDAVADDR Register Field Descriptions .....	165
3-26.	NMCPUWRAVADDR Register Field Descriptions .....	166
3-27.	NMCPUFAVADDR Register Field Descriptions .....	167
3-28.	NMDMAWRAVADDR Register Field Descriptions .....	168
3-29.	NMCLA1RDAVADDR Register Field Descriptions .....	169
3-30.	NMCLA1WRAVADDR Register Field Descriptions .....	170
3-31.	NMCLA1FAVADDR Register Field Descriptions .....	171
3-32.	MAVFLG Register Field Descriptions .....	172
3-33.	MAVSET Register Field Descriptions .....	173
3-34.	MAVCLR Register Field Descriptions .....	174
3-35.	MAVINTEN Register Field Descriptions .....	175
3-36.	MCPUFAVADDR Register Field Descriptions .....	176
3-37.	MCPUWRAVADDR Register Field Descriptions .....	177
3-38.	MDMAWRAVADDR Register Field Descriptions .....	178
3-39.	CLK_CFG_REGS Registers .....	179
3-40.	CLK_CFG_REGS Access Type Codes.....	179
3-41.	CLKCFGLOCK1 Register Field Descriptions .....	180
3-42.	CLKSRCCTL1 Register Field Descriptions.....	182
3-43.	CLKSRCCTL2 Register Field Descriptions.....	184

3-44.	CLKSRCCTL3 Register Field Descriptions.....	185
3-45.	SYSPLLCTL1 Register Field Descriptions .....	186
3-46.	SYSPLLMULT Register Field Descriptions.....	187
3-47.	SYSPLLSTS Register Field Descriptions.....	188
3-48.	SYSCLKDIVSEL Register Field Descriptions .....	189
3-49.	XCLKOUTDIVSEL Register Field Descriptions.....	190
3-50.	LOSPCP Register Field Descriptions .....	191
3-51.	MCDCCR Register Field Descriptions .....	192
3-52.	X1CNT Register Field Descriptions .....	193
3-53.	XTALCR Register Field Descriptions .....	194
3-54.	CPU_SYS_REGS Registers .....	195
3-55.	CPU_SYS_REGS Access Type Codes.....	195
3-56.	CPUSYSLOCK1 Register Field Descriptions .....	197
3-57.	PIEVERRADDR Register Field Descriptions.....	200
3-58.	PCLKCR0 Register Field Descriptions.....	201
3-59.	PCLKCR2 Register Field Descriptions.....	203
3-60.	PCLKCR3 Register Field Descriptions.....	205
3-61.	PCLKCR4 Register Field Descriptions.....	207
3-62.	PCLKCR6 Register Field Descriptions.....	208
3-63.	PCLKCR7 Register Field Descriptions.....	209
3-64.	PCLKCR8 Register Field Descriptions.....	210
3-65.	PCLKCR9 Register Field Descriptions.....	211
3-66.	PCLKCR10 Register Field Descriptions .....	212
3-67.	PCLKCR13 Register Field Descriptions .....	213
3-68.	PCLKCR14 Register Field Descriptions .....	214
3-69.	PCLKCR15 Register Field Descriptions .....	216
3-70.	PCLKCR16 Register Field Descriptions .....	218
3-71.	PCLKCR17 Register Field Descriptions .....	219
3-72.	PCLKCR18 Register Field Descriptions .....	220
3-73.	PCLKCR19 Register Field Descriptions .....	221
3-74.	PCLKCR20 Register Field Descriptions .....	222
3-75.	PCLKCR21 Register Field Descriptions .....	223
3-76.	LPMCR Register Field Descriptions.....	224
3-77.	GPIOLPMSEL0 Register Field Descriptions .....	225
3-78.	GPIOLPMSEL1 Register Field Descriptions .....	228
3-79.	TMR2CLKCTL Register Field Descriptions .....	231
3-80.	RESCCLR Register Field Descriptions .....	232
3-81.	RESC Register Field Descriptions.....	234
3-82.	CPUTIMER_REGS Registers .....	236
3-83.	CPUTIMER_REGS Access Type Codes .....	236
3-84.	TIM Register Field Descriptions .....	237
3-85.	PRD Register Field Descriptions .....	238
3-86.	TCR Register Field Descriptions.....	239
3-87.	TPR Register Field Descriptions.....	241
3-88.	TPRH Register Field Descriptions .....	242
3-89.	DCSM_BANK0_Z1_REGS Registers .....	243
3-90.	DCSM_BANK0_Z1_REGS Access Type Codes .....	243
3-91.	B0_Z1_LINKPOINTER Register Field Descriptions.....	244
3-92.	Z1_OTPSECLOCK Register Field Descriptions .....	245



3-93.	Z1_BOOTDEF_HIGH Register Field Descriptions .....	246
3-94.	B0_Z1_LINKPOINTERERR Register Field Descriptions .....	247
3-95.	Z1_BOOTPIN_CONFIG Register Field Descriptions .....	248
3-96.	Z1_GPREG2 Register Field Descriptions .....	249
3-97.	Z1_BOOTDEF_LOW Register Field Descriptions .....	250
3-98.	Z1_CSMKEY0 Register Field Descriptions.....	251
3-99.	Z1_CSMKEY1 Register Field Descriptions.....	252
3-100.	Z1_CSMKEY2 Register Field Descriptions.....	253
3-101.	Z1_CSMKEY3 Register Field Descriptions.....	254
3-102.	Z1_CR Register Field Descriptions.....	255
3-103.	B0_Z1_GRABSECTR Register Field Descriptions .....	256
3-104.	Z1_GRABRAMR Register Field Descriptions .....	260
3-105.	B0_Z1_EXEONLYSECTR Register Field Descriptions.....	262
3-106.	Z1_EXEONLYRAMR Register Field Descriptions.....	266
3-107.	DCSM_BANK0_Z2_REGS Registers .....	268
3-108.	DCSM_BANK0_Z2_REGS Access Type Codes .....	268
3-109.	B0_Z2_LINKPOINTER Register Field Descriptions.....	269
3-110.	Z2_OTPSECLOCK Register Field Descriptions .....	270
3-111.	B0_Z2_LINKPOINTERERR Register Field Descriptions .....	271
3-112.	Z2_CSMKEY0 Register Field Descriptions.....	272
3-113.	Z2_CSMKEY1 Register Field Descriptions.....	273
3-114.	Z2_CSMKEY2 Register Field Descriptions.....	274
3-115.	Z2_CSMKEY3 Register Field Descriptions.....	275
3-116.	Z2_CR Register Field Descriptions.....	276
3-117.	B0_Z2_GRABSECTR Register Field Descriptions .....	277
3-118.	Z2_GRABRAMR Register Field Descriptions .....	281
3-119.	B0_Z2_EXEONLYSECTR Register Field Descriptions .....	283
3-120.	Z2_EXEONLYRAMR Register Field Descriptions .....	287
3-121.	DCSM_BANK1_Z1_REGS Registers .....	289
3-122.	DCSM_BANK1_Z1_REGS Access Type Codes .....	289
3-123.	B1_Z1_LINKPOINTER Register Field Descriptions.....	290
3-124.	B1_Z1_LINKPOINTERERR Register Field Descriptions .....	291
3-125.	B1_Z1_GRABSECTR Register Field Descriptions .....	292
3-126.	B1_Z1_EXEONLYSECTR Register Field Descriptions .....	296
3-127.	DCSM_BANK1_Z2_REGS Registers .....	300
3-128.	DCSM_BANK1_Z2_REGS Access Type Codes .....	300
3-129.	B1_Z2_LINKPOINTER Register Field Descriptions.....	301
3-130.	B1_Z2_LINKPOINTERERR Register Field Descriptions .....	302
3-131.	B1_Z2_GRABSECTR Register Field Descriptions .....	303
3-132.	B1_Z2_EXEONLYSECTR Register Field Descriptions .....	307
3-133.	DCSM_COMMON_REGS Registers .....	311
3-134.	DCSM_COMMON_REGS Access Type Codes .....	311
3-135.	FLSEM Register Field Descriptions .....	312
3-136.	B0_SECTSTAT Register Field Descriptions .....	313
3-137.	RAMSTAT Register Field Descriptions .....	316
3-138.	B1_SECTSTAT Register Field Descriptions .....	318
3-139.	SECERRSTAT Register Field Descriptions .....	321
3-140.	SECERRCLR Register Field Descriptions .....	322
3-141.	SECERRFRC Register Field Descriptions .....	323

3-142. DEV_CFG_REGS Registers .....	324
3-143. DEV_CFG_REGS Access Type Codes .....	324
3-144. PARTIDL Register Field Descriptions .....	326
3-145. PARTIDH Register Field Descriptions .....	328
3-146. REVID Register Field Descriptions .....	329
3-147. DC21 Register Field Descriptions .....	330
3-148. FUSEERR Register Field Descriptions .....	331
3-149. SOFTPRES0 Register Field Descriptions .....	332
3-150. SOFTPRES2 Register Field Descriptions .....	333
3-151. SOFTPRES3 Register Field Descriptions .....	335
3-152. SOFTPRES4 Register Field Descriptions .....	336
3-153. SOFTPRES6 Register Field Descriptions .....	337
3-154. SOFTPRES7 Register Field Descriptions .....	338
3-155. SOFTPRES8 Register Field Descriptions .....	339
3-156. SOFTPRES9 Register Field Descriptions .....	340
3-157. SOFTPRES10 Register Field Descriptions.....	341
3-158. SOFTPRES13 Register Field Descriptions.....	342
3-159. SOFTPRES14 Register Field Descriptions.....	343
3-160. SOFTPRES15 Register Field Descriptions.....	344
3-161. SOFTPRES16 Register Field Descriptions.....	345
3-162. SOFTPRES17 Register Field Descriptions.....	346
3-163. SOFTPRES18 Register Field Descriptions.....	347
3-164. SOFTPRES19 Register Field Descriptions.....	348
3-165. SOFTPRES20 Register Field Descriptions.....	349
3-166. SOFTPRES21 Register Field Descriptions.....	350
3-167. TAP_STATUS Register Field Descriptions.....	351
3-168. DMA_CLA_SRC_SEL_REGS Registers .....	352
3-169. DMA_CLA_SRC_SEL_REGS Access Type Codes.....	352
3-170. CLA1TASKSRCSELLOCK Register Field Descriptions .....	353
3-171. DMACHSRCSELLOCK Register Field Descriptions .....	354
3-172. CLA1TASKSRCSEL1 Register Field Descriptions .....	355
3-173. CLA1TASKSRCSEL2 Register Field Descriptions .....	356
3-174. DMACHSRCSEL1 Register Field Descriptions .....	357
3-175. DMACHSRCSEL2 Register Field Descriptions .....	358
3-176. FLASH_CTRL_REGS Registers.....	359
3-177. FLASH_CTRL_REGS Access Type Codes .....	359
3-178. FRDCNTL Register Field Descriptions.....	360
3-179. FBAC Register Field Descriptions .....	361
3-180. FBFALLBACK Register Field Descriptions .....	362
3-181. FBPRDY Register Field Descriptions .....	363
3-182. FPAC1 Register Field Descriptions .....	364
3-183. FPAC2 Register Field Descriptions .....	365
3-184. FMSTAT Register Field Descriptions .....	366
3-185. FRD_INTF_CTRL Register Field Descriptions.....	368
3-186. FLASH_ECC_REGS Registers .....	369
3-187. FLASH_ECC_REGS Access Type Codes .....	369
3-188. ECC_ENABLE Register Field Descriptions .....	371
3-189. SINGLE_ERR_ADDR_LOW Register Field Descriptions .....	372
3-190. SINGLE_ERR_ADDR_HIGH Register Field Descriptions.....	373

3-191. UNC_ERR_ADDR_LOW Register Field Descriptions .....	374
3-192. UNC_ERR_ADDR_HIGH Register Field Descriptions.....	375
3-193. ERR_STATUS Register Field Descriptions .....	376
3-194. ERR_POS Register Field Descriptions .....	378
3-195. ERR_STATUS_CLR Register Field Descriptions .....	379
3-196. ERR_CNT Register Field Descriptions .....	380
3-197. ERR_THRESHOLD Register Field Descriptions .....	381
3-198. ERR_INTFLG Register Field Descriptions .....	382
3-199. ERR_INTCLR Register Field Descriptions .....	383
3-200. FDATAH_TEST Register Field Descriptions .....	384
3-201. FDATAL_TEST Register Field Descriptions .....	385
3-202. FADDR_TEST Register Field Descriptions .....	386
3-203. FECC_TEST Register Field Descriptions .....	387
3-204. FECC_CTRL Register Field Descriptions .....	388
3-205. FOUTH_TEST Register Field Descriptions .....	389
3-206. FOUTL_TEST Register Field Descriptions .....	390
3-207. FECC_STATUS Register Field Descriptions.....	391
3-208. MEM_CFG_REGS Registers .....	392
3-209. MEM_CFG_REGS Access Type Codes.....	392
3-210. DxLOCK Register Field Descriptions .....	394
3-211. DxCOMMIT Register Field Descriptions .....	395
3-212. DxACCPROT0 Register Field Descriptions .....	396
3-213. DxTEST Register Field Descriptions .....	397
3-214. DxINIT Register Field Descriptions.....	398
3-215. DxINITDONE Register Field Descriptions .....	399
3-216. LSxLOCK Register Field Descriptions .....	400
3-217. LSxCOMMIT Register Field Descriptions.....	402
3-218. LSxMSEL Register Field Descriptions .....	404
3-219. LSxCLAPGM Register Field Descriptions .....	406
3-220. LSxACCPROT0 Register Field Descriptions.....	408
3-221. LSxACCPROT1 Register Field Descriptions.....	410
3-222. LSxTEST Register Field Descriptions.....	412
3-223. LSxINIT Register Field Descriptions .....	414
3-224. LSxINITDONE Register Field Descriptions.....	416
3-225. GSxLOCK Register Field Descriptions.....	418
3-226. GSxCOMMIT Register Field Descriptions .....	420
3-227. GSxACCPROT0 Register Field Descriptions .....	422
3-228. GSxACCPROT1 Register Field Descriptions .....	424
3-229. GSxACCPROT2 Register Field Descriptions .....	425
3-230. GSxACCPROT3 Register Field Descriptions .....	426
3-231. GSxTEST Register Field Descriptions .....	427
3-232. GSxINIT Register Field Descriptions .....	429
3-233. GSxINITDONE Register Field Descriptions .....	431
3-234. MSGxLOCK Register Field Descriptions .....	433
3-235. MSGxCOMMIT Register Field Descriptions.....	434
3-236. MSGxTEST Register Field Descriptions.....	435
3-237. MSGxINIT Register Field Descriptions.....	436
3-238. MSGxINITDONE Register Field Descriptions.....	437
3-239. MEMORY_ERROR_REGS Registers.....	438

3-240. MEMORY_ERROR_REGS Access Type Codes .....	438
3-241. UCERRFLG Register Field Descriptions .....	439
3-242. UCERRSET Register Field Descriptions .....	440
3-243. UCERRCLR Register Field Descriptions .....	441
3-244. UCCPUREADDR Register Field Descriptions .....	442
3-245. UCDMAREADDR Register Field Descriptions .....	443
3-246. UCCLA1READDR Register Field Descriptions .....	444
3-247. CERRFLG Register Field Descriptions .....	445
3-248. CERRSET Register Field Descriptions .....	446
3-249. CERRCLR Register Field Descriptions .....	447
3-250. CCPUREADDR Register Field Descriptions .....	448
3-251. CERRCNT Register Field Descriptions .....	449
3-252. CERRTHRES Register Field Descriptions .....	450
3-253. CEINTFLG Register Field Descriptions .....	451
3-254. CEINTCLR Register Field Descriptions .....	452
3-255. CEINTSET Register Field Descriptions .....	453
3-256. CEINTEN Register Field Descriptions .....	454
3-257. NMI_INTRUPT_REGS Registers .....	455
3-258. NMI_INTRUPT_REGS Access Type Codes .....	455
3-259. NMICFG Register Field Descriptions .....	456
3-260. NMIFLG Register Field Descriptions .....	457
3-261. NMIFLGCLR Register Field Descriptions .....	459
3-262. NMIFLGFRC Register Field Descriptions .....	460
3-263. NMIWDCNT Register Field Descriptions .....	461
3-264. NMIWDPRD Register Field Descriptions .....	462
3-265. NMISHDFLG Register Field Descriptions .....	463
3-266. PERIPH_AC_REGS Registers .....	464
3-267. PERIPH_AC_REGS Access Type Codes .....	465
3-268. ADCA_AC Register Field Descriptions .....	466
3-269. ADCB_AC Register Field Descriptions .....	467
3-270. ADCC_AC Register Field Descriptions .....	468
3-271. CMPSS1_AC Register Field Descriptions .....	469
3-272. CMPSS2_AC Register Field Descriptions .....	470
3-273. CMPSS3_AC Register Field Descriptions .....	471
3-274. CMPSS4_AC Register Field Descriptions .....	472
3-275. CMPSS5_AC Register Field Descriptions .....	473
3-276. CMPSS6_AC Register Field Descriptions .....	474
3-277. CMPSS7_AC Register Field Descriptions .....	475
3-278. DACA_AC Register Field Descriptions .....	476
3-279. DACB_AC Register Field Descriptions .....	477
3-280. PGA1_AC Register Field Descriptions .....	478
3-281. PGA2_AC Register Field Descriptions .....	479
3-282. PGA3_AC Register Field Descriptions .....	480
3-283. PGA4_AC Register Field Descriptions .....	481
3-284. PGA5_AC Register Field Descriptions .....	482
3-285. PGA6_AC Register Field Descriptions .....	483
3-286. PGA7_AC Register Field Descriptions .....	484
3-287. EPWM1_AC Register Field Descriptions .....	485
3-288. EPWM2_AC Register Field Descriptions .....	486

3-289. EPWM3_AC Register Field Descriptions .....	487
3-290. EPWM4_AC Register Field Descriptions .....	488
3-291. EPWM5_AC Register Field Descriptions .....	489
3-292. EPWM6_AC Register Field Descriptions .....	490
3-293. EPWM7_AC Register Field Descriptions .....	491
3-294. EPWM8_AC Register Field Descriptions .....	492
3-295. EQEP1_AC Register Field Descriptions .....	493
3-296. EQEP2_AC Register Field Descriptions .....	494
3-297. ECAP1_AC Register Field Descriptions .....	495
3-298. ECAP2_AC Register Field Descriptions .....	496
3-299. ECAP3_AC Register Field Descriptions .....	497
3-300. ECAP4_AC Register Field Descriptions .....	498
3-301. ECAP5_AC Register Field Descriptions .....	499
3-302. ECAP6_AC Register Field Descriptions .....	500
3-303. ECAP7_AC Register Field Descriptions .....	501
3-304. SDFM1_AC Register Field Descriptions .....	502
3-305. CLB1_AC Register Field Descriptions .....	503
3-306. CLB2_AC Register Field Descriptions .....	504
3-307. CLB3_AC Register Field Descriptions .....	505
3-308. CLB4_AC Register Field Descriptions .....	506
3-309. CLA1PROMCRC_AC Register Field Descriptions .....	507
3-310. SPIA_AC Register Field Descriptions .....	508
3-311. SPIB_AC Register Field Descriptions .....	509
3-312. PMBUS_A_AC Register Field Descriptions .....	510
3-313. LIN_A_AC Register Field Descriptions .....	511
3-314. DCANA_AC Register Field Descriptions .....	512
3-315. DCANB_AC Register Field Descriptions .....	513
3-316. FSIATX_AC Register Field Descriptions .....	514
3-317. FSIARX_AC Register Field Descriptions .....	515
3-318. HRPWM_A_AC Register Field Descriptions .....	516
3-319. PERIPH_AC_LOCK Register Field Descriptions .....	517
3-320. PIE_CTRL_REGS Registers .....	518
3-321. PIE_CTRL_REGS Access Type Codes .....	518
3-322. PIECTRL Register Field Descriptions .....	520
3-323. PIEACK Register Field Descriptions .....	521
3-324. PIEIER1 Register Field Descriptions .....	522
3-325. PIEIFR1 Register Field Descriptions .....	524
3-326. PIEIER2 Register Field Descriptions .....	526
3-327. PIEIFR2 Register Field Descriptions .....	528
3-328. PIEIER3 Register Field Descriptions .....	530
3-329. PIEIFR3 Register Field Descriptions .....	532
3-330. PIEIER4 Register Field Descriptions .....	534
3-331. PIEIFR4 Register Field Descriptions .....	536
3-332. PIEIER5 Register Field Descriptions .....	538
3-333. PIEIFR5 Register Field Descriptions .....	540
3-334. PIEIER6 Register Field Descriptions .....	542
3-335. PIEIFR6 Register Field Descriptions .....	544
3-336. PIEIER7 Register Field Descriptions .....	546
3-337. PIEIFR7 Register Field Descriptions .....	548

3-338. PIEIER8 Register Field Descriptions .....	550
3-339. PIEIFR8 Register Field Descriptions .....	552
3-340. PIEIER9 Register Field Descriptions .....	554
3-341. PIEIFR9 Register Field Descriptions .....	556
3-342. PIEIER10 Register Field Descriptions .....	558
3-343. PIEIFR10 Register Field Descriptions .....	560
3-344. PIEIER11 Register Field Descriptions .....	562
3-345. PIEIFR11 Register Field Descriptions .....	564
3-346. PIEIER12 Register Field Descriptions .....	566
3-347. PIEIFR12 Register Field Descriptions .....	568
3-348. ROM_PREFETCH_REGS Registers.....	570
3-349. ROM_PREFETCH_REGS Access Type Codes .....	570
3-350. ROM_WAIT_STATE_REGS Registers .....	571
3-351. ROM_WAIT_STATE_REGS Access Type Codes .....	571
3-352. UID_REGS Registers .....	572
3-353. UID_REGS Access Type Codes.....	572
3-354. UID_PSRAND0 Register Field Descriptions .....	573
3-355. UID_PSRAND1 Register Field Descriptions .....	574
3-356. UID_PSRAND2 Register Field Descriptions .....	575
3-357. UID_PSRAND3 Register Field Descriptions .....	576
3-358. UID_PSRAND4 Register Field Descriptions .....	577
3-359. UID_PSRAND5 Register Field Descriptions .....	578
3-360. UID_UNIQUE Register Field Descriptions .....	579
3-361. UID_CHECKSUM Register Field Descriptions.....	580
3-362. WD_REGS Registers .....	581
3-363. WD_REGS Access Type Codes.....	581
3-364. SCSR Register Field Descriptions.....	582
3-365. WDCNTR Register Field Descriptions .....	583
3-366. WDKEY Register Field Descriptions .....	584
3-367. WDCR Register Field Descriptions.....	585
3-368. WDWCR Register Field Descriptions .....	587
3-369. XINT_REGS Registers.....	588
3-370. XINT_REGS Access Type Codes .....	588
3-371. XINT1CR Register Field Descriptions.....	589
3-372. XINT2CR Register Field Descriptions.....	590
3-373. XINT3CR Register Field Descriptions.....	591
3-374. XINT4CR Register Field Descriptions.....	592
3-375. XINT5CR Register Field Descriptions.....	593
3-376. XINT1CTR Register Field Descriptions .....	594
3-377. XINT2CTR Register Field Descriptions .....	595
3-378. XINT3CTR Register Field Descriptions .....	596
3-379. CPUTIMER Registers to Driverlib Functions.....	597
3-380. ASYSCTL Registers to Driverlib Functions.....	597
3-381. PIE Registers to Driverlib Functions .....	598
3-382. SYSCTL Registers to Driverlib Functions .....	599
3-383. NMI Registers to Driverlib Functions .....	603
3-384. XINT Registers to Driverlib Functions.....	603
3-385. DCSM Registers to Driverlib Functions .....	603
3-386. MEMCFG Registers to Driverlib Functions .....	605



4-1.	ROM Memory .....	609
4-2.	Boot ROM Philosophy.....	609
4-3.	Device Default Boot Modes .....	610
4-4.	All Available Boot Modes .....	610
4-5.	BOOTPIN_CONFIG Bit Fields .....	610
4-6.	Standalone Boot Mode Select Pin Decoding .....	611
4-7.	BOOTDEF Bit Fields .....	612
4-8.	Zero Boot Pin Boot Table Result .....	612
4-9.	One Boot Pin Boot Table Result.....	613
4-10.	Boot ROM Reset Causes and Actions.....	617
4-11.	Boot ROM Exceptions and Actions.....	617
4-12.	Boot ROM Registers .....	618
4-13.	User-Configurable DCSM OTP Fields .....	618
4-14.	Entry Point Addresses .....	618
4-15.	Wait Point Addresses .....	619
4-16.	Boot ROM Memory Map (Silicon revision 0,A) .....	619
4-17.	Boot ROM Memory Map (Silicon revision B).....	619
4-18.	CLA Data ROM Memory Map.....	620
4-19.	Reserved RAM and Flash Memory Map (Silicon revision A) .....	620
4-20.	Reserved RAM and Flash Memory Map (Silicon revision B) .....	620
4-21.	CLA ROM Tables .....	621
4-22.	SPI 8-Bit Data Stream .....	624
4-23.	I2C 8-Bit Data Stream .....	628
4-24.	Parallel GPIO Boot 8-Bit Data Stream .....	629
4-25.	Bit-Rate Value for Internal Oscillators.....	633
4-26.	CAN 8-Bit Data Stream .....	633
4-27.	LSB/MSB Loading Sequence in 8-Bit Data Stream .....	635
4-28.	SCI Boot Options .....	636
4-29.	CAN Boot Options .....	636
4-30.	Flash Boot Options .....	636
4-31.	Wait Boot Options .....	637
4-32.	SPI Boot Options .....	637
4-33.	I2C Boot Options.....	637
4-34.	Parallel Boot Options.....	637
4-35.	RAM Boot Options.....	637
4-36.	DCSM Z1-OTP-BOOT-GPREG2 Bit Fields .....	638
4-37.	Boot Clock Sources .....	638
4-38.	Clock State after Boot ROM.....	638
4-39.	Boot Status Address.....	639
4-40.	Boot Status Bit Fields .....	639
4-41.	Flash Single Bit Error Status Addresses.....	639
4-42.	Boot ROM Version Information .....	640
4-43.	Boot Loader Options .....	640
5-1.	Configuration Options .....	647
5-2.	Pipeline Behavior of the MDEBUGSTOP1 Instruction.....	653
5-3.	Write Followed by Read - Read Occurs First .....	658
5-4.	Write Followed by Read - Write Occurs First .....	658
5-5.	ADC to CLA Early Interrupt Response .....	660
5-6.	Operand Nomenclature .....	663



5-7.	INSTRUCTION dest, source1, source2 Short Description .....	664
5-8.	Addressing Modes .....	665
5-9.	Shift Field Encoding .....	666
5-10.	Operand Encoding.....	666
5-11.	Condition Field Encoding .....	666
5-12.	General Instructions .....	667
5-13.	Pipeline Activity For MBCNDD, Branch Not Taken .....	682
5-14.	Pipeline Activity For MBCNDD, Branch Taken .....	682
5-15.	Pipeline Activity For MCCNDD, Call Not Taken .....	688
5-16.	Pipeline Activity For MCCNDD, Call Taken .....	688
5-17.	Pipeline Activity For MMOV16 MARx, MRa , #16l .....	722
5-18.	Pipeline Activity For MMOV16 MAR0/MAR1, mem16 .....	725
5-19.	Pipeline Activity For MMOVI16 MAR0/MAR1, #16l .....	739
5-20.	Pipeline Activity For MRCNDD, Return Not Taken .....	761
5-21.	Pipeline Activity For MRCNDD, Return Taken .....	761
5-22.	Pipeline Activity For MSTOP .....	766
5-23.	CLA Base Address Table.....	781
5-24.	CLA_ONLY_REGS Registers .....	782
5-25.	CLA_ONLY_REGS Access Type Codes .....	782
5-26.	_MVECTBGRNDACTIVE Register Field Descriptions.....	783
5-27.	_MPSACTL Register Field Descriptions .....	784
5-28.	_MPSA1 Register Field Descriptions .....	786
5-29.	_MPSA2 Register Field Descriptions .....	787
5-30.	SOFTINTEN Register Field Descriptions.....	788
5-31.	SOFTINTFRC Register Field Descriptions .....	790
5-32.	CLA_SOFTINT_REGS Registers.....	791
5-33.	CLA_SOFTINT_REGS Access Type Codes .....	791
5-34.	SOFTINTEN Register Field Descriptions.....	792
5-35.	SOFTINTFRC Register Field Descriptions .....	794
5-36.	CLA_REGS Registers.....	795
5-37.	CLA_REGS Access Type Codes .....	795
5-38.	MVECT1 Register Field Descriptions .....	797
5-39.	MVECT2 Register Field Descriptions .....	798
5-40.	MVECT3 Register Field Descriptions .....	799
5-41.	MVECT4 Register Field Descriptions .....	800
5-42.	MVECT5 Register Field Descriptions .....	801
5-43.	MVECT6 Register Field Descriptions .....	802
5-44.	MVECT7 Register Field Descriptions .....	803
5-45.	MVECT8 Register Field Descriptions .....	804
5-46.	MCTL Register Field Descriptions.....	805
5-47.	_MVECTBGRNDACTIVE Register Field Descriptions.....	806
5-48.	SOFTINTEN Register Field Descriptions .....	807
5-49.	_MSTSBGRND Register Field Descriptions .....	809
5-50.	_MCTLBGRND Register Field Descriptions.....	810
5-51.	_MVECTBGRND Register Field Descriptions .....	811
5-52.	MIFR Register Field Descriptions.....	812
5-53.	MIOVF Register Field Descriptions.....	817
5-54.	MIFRC Register Field Descriptions.....	820
5-55.	MICLR Register Field Descriptions.....	822

5-56.	MICLROVF Register Field Descriptions .....	824
5-57.	MIER Register Field Descriptions .....	826
5-58.	MIRUN Register Field Descriptions .....	829
5-59.	_MPC Register Field Descriptions.....	832
5-60.	_MAR0 Register Field Descriptions .....	833
5-61.	_MAR1 Register Field Descriptions .....	834
5-62.	_MSTF Register Field Descriptions .....	835
5-63.	_MR0 Register Field Descriptions .....	838
5-64.	_MR1 Register Field Descriptions .....	839
5-65.	_MR2 Register Field Descriptions .....	840
5-66.	_MR3 Register Field Descriptions .....	841
5-67.	_MPSACTL Register Field Descriptions .....	842
5-68.	_MPSA1 Register Field Descriptions .....	844
5-69.	_MPSA2 Register Field Descriptions .....	845
6-1.	DCC Base Address Table .....	850
6-2.	DCC_REGS Registers .....	851
6-3.	DCC_REGS Access Type Codes .....	851
6-4.	DCCGCTRL Register Field Descriptions .....	852
6-5.	DCCREV Register Field Descriptions.....	853
6-6.	DCCNTSEED0 Register Field Descriptions .....	854
6-7.	DCCVALIDSEED0 Register Field Descriptions.....	855
6-8.	DCCNTSEED1 Register Field Descriptions .....	856
6-9.	DCCSTATUS Register Field Descriptions.....	857
6-10.	DCCNT0 Register Field Descriptions .....	858
6-11.	DCCVALID0 Register Field Descriptions .....	859
6-12.	DCCNT1 Register Field Descriptions .....	860
6-13.	DCCCLKSRC1 Register Field Descriptions .....	861
6-14.	DCCCLKSRC0 Register Field Descriptions .....	862
6-15.	DCC Registers to Driverlib Functions .....	863
7-1.	CLA PROM CRC Base Address Table .....	867
7-2.	CLA_PROM_CRC32_REGS Registers.....	868
7-3.	CLA_PROM_CRC32_REGS Access Type Codes .....	868
7-4.	CRC32_CONTROLREG Register Field Descriptions.....	869
7-5.	CRC32_STARTADDRESS Register Field Descriptions .....	871
7-6.	CRC32_SEED Register Field Descriptions .....	872
7-7.	CRC32_STATUSREG Register Field Descriptions .....	873
7-8.	CRC32_CRCRESULT Register Field Descriptions .....	874
7-9.	CRC32_GOLDENCRC Register Field Descriptions.....	875
7-10.	CRC32_INTEN Register Field Descriptions.....	876
7-11.	CRC32_FLG Register Field Descriptions.....	877
7-12.	CRC32_CLR Register Field Descriptions .....	878
7-13.	CRC32_FRC Register Field Descriptions .....	879
8-1.	Sampling Period .....	885
8-2.	Sampling Frequency .....	885
8-3.	Case 1: Three-Sample Sampling Window Width .....	886
8-4.	Case 2: Six-Sample Sampling Window Width .....	886
8-5.	GPIO Muxed Pins.....	888
8-6.	GPIO and Peripheral Muxing.....	890
8-7.	Peripheral Muxing (multiple pins assigned) .....	892

8-8.	GPIO Base Address Table .....	894
8-9.	GPIO_CTRL_REGS Registers.....	895
8-10.	GPIO_CTRL_REGS Access Type Codes .....	896
8-11.	GPACTRL Register Field Descriptions .....	897
8-12.	GPAQSEL1 Register Field Descriptions.....	898
8-13.	GPAQSEL2 Register Field Descriptions.....	900
8-14.	GPAMUX1 Register Field Descriptions .....	902
8-15.	GPAMUX2 Register Field Descriptions .....	904
8-16.	GPADIR Register Field Descriptions.....	906
8-17.	GPAPUD Register Field Descriptions.....	908
8-18.	GPAINV Register Field Descriptions .....	910
8-19.	GPAODR Register Field Descriptions .....	912
8-20.	GPAAMSEL Register Field Descriptions .....	914
8-21.	GPAGMUX1 Register Field Descriptions.....	916
8-22.	GPAGMUX2 Register Field Descriptions.....	918
8-23.	GPACSEL1 Register Field Descriptions .....	920
8-24.	GPACSEL2 Register Field Descriptions.....	921
8-25.	GPACSEL3 Register Field Descriptions .....	922
8-26.	GPACSEL4 Register Field Descriptions .....	923
8-27.	GPALOCK Register Field Descriptions .....	924
8-28.	GPACR Register Field Descriptions.....	926
8-29.	GPBCTRL Register Field Descriptions .....	928
8-30.	GPBQSEL1 Register Field Descriptions.....	929
8-31.	GPBQSEL2 Register Field Descriptions.....	931
8-32.	GPBMUX1 Register Field Descriptions .....	932
8-33.	GPBMUX2 Register Field Descriptions .....	933
8-34.	GPBDIR Register Field Descriptions.....	934
8-35.	GPBPUD Register Field Descriptions.....	936
8-36.	GPBINV Register Field Descriptions .....	938
8-37.	GPBODR Register Field Descriptions .....	940
8-38.	GPBGMUX1 Register Field Descriptions.....	942
8-39.	GPBGMUX2 Register Field Descriptions.....	943
8-40.	GPBCSEL1 Register Field Descriptions .....	944
8-41.	GPBCSEL2 Register Field Descriptions .....	945
8-42.	GPBCSEL3 Register Field Descriptions .....	946
8-43.	GPBCSEL4 Register Field Descriptions .....	947
8-44.	GPBLOCK Register Field Descriptions .....	948
8-45.	GPBCR Register Field Descriptions.....	950
8-46.	GPHCTRL Register Field Descriptions .....	952
8-47.	GPHQSEL1 Register Field Descriptions.....	953
8-48.	GPHQSEL2 Register Field Descriptions.....	955
8-49.	GPHPUD Register Field Descriptions.....	957
8-50.	GPHINV Register Field Descriptions.....	959
8-51.	GPHAMSEL Register Field Descriptions .....	961
8-52.	GPHLOCK Register Field Descriptions .....	963
8-53.	GPHCR Register Field Descriptions .....	965
8-54.	GPIO_DATA_REGS Registers .....	967
8-55.	GPIO_DATA_REGS Access Type Codes.....	967
8-56.	GPADAT Register Field Descriptions .....	968

8-57.	GPASET Register Field Descriptions .....	970
8-58.	GPACLEAR Register Field Descriptions .....	972
8-59.	GPATOGGLE Register Field Descriptions .....	974
8-60.	GPBDAT Register Field Descriptions .....	976
8-61.	GPBSET Register Field Descriptions .....	978
8-62.	GPBCLEAR Register Field Descriptions .....	980
8-63.	GPBTOGGLE Register Field Descriptions .....	982
8-64.	GPHDAT Register Field Descriptions .....	984
8-65.	GPIO Registers to Driverlib Functions .....	986
9-1.	Input X-BAR Destinations.....	991
9-2.	ePWM X-BAR Mux Configuration Table .....	993
9-3.	CLB X-BAR Mux Configuration Table.....	994
9-4.	Output X-BAR Mux Configuration Table.....	997
9-5.	System Control Base Address Table.....	999
9-6.	XBAR_REGS Registers .....	1000
9-7.	XBAR_REGS Access Type Codes .....	1000
9-8.	XBARFLG1 Register Field Descriptions.....	1001
9-9.	XBARFLG2 Register Field Descriptions.....	1007
9-10.	XBARFLG3 Register Field Descriptions.....	1013
9-11.	XBARFLG4 Register Field Descriptions.....	1019
9-12.	XBARCLR1 Register Field Descriptions .....	1024
9-13.	XBARCLR2 Register Field Descriptions .....	1027
9-14.	XBARCLR3 Register Field Descriptions .....	1030
9-15.	XBARCLR4 Register Field Descriptions .....	1033
9-16.	INPUT_XBAR_REGS Registers .....	1036
9-17.	INPUT_XBAR_REGS Access Type Codes.....	1036
9-18.	INPUT1SELECT Register Field Descriptions.....	1037
9-19.	INPUT2SELECT Register Field Descriptions.....	1038
9-20.	INPUT3SELECT Register Field Descriptions.....	1039
9-21.	INPUT4SELECT Register Field Descriptions.....	1040
9-22.	INPUT5SELECT Register Field Descriptions.....	1041
9-23.	INPUT6SELECT Register Field Descriptions.....	1042
9-24.	INPUT7SELECT Register Field Descriptions.....	1043
9-25.	INPUT8SELECT Register Field Descriptions.....	1044
9-26.	INPUT9SELECT Register Field Descriptions.....	1045
9-27.	INPUT10SELECT Register Field Descriptions .....	1046
9-28.	INPUT11SELECT Register Field Descriptions .....	1047
9-29.	INPUT12SELECT Register Field Descriptions .....	1048
9-30.	INPUT13SELECT Register Field Descriptions .....	1049
9-31.	INPUT14SELECT Register Field Descriptions .....	1050
9-32.	INPUT15SELECT Register Field Descriptions .....	1051
9-33.	INPUT16SELECT Register Field Descriptions .....	1052
9-34.	INPUTSELECTLOCK Register Field Descriptions.....	1053
9-35.	OUTPUT_XBAR_REGS Registers .....	1055
9-36.	OUTPUT_XBAR_REGS Access Type Codes.....	1055
9-37.	OUTPUT1MUX0TO15CFG Register Field Descriptions .....	1057
9-38.	OUTPUT1MUX16TO31CFG Register Field Descriptions.....	1060
9-39.	OUTPUT2MUX0TO15CFG Register Field Descriptions .....	1063
9-40.	OUTPUT2MUX16TO31CFG Register Field Descriptions.....	1066

9-41.	OUTPUT3MUX0TO15CFG Register Field Descriptions .....	1069
9-42.	OUTPUT3MUX16TO31CFG Register Field Descriptions.....	1072
9-43.	OUTPUT4MUX0TO15CFG Register Field Descriptions .....	1075
9-44.	OUTPUT4MUX16TO31CFG Register Field Descriptions.....	1078
9-45.	OUTPUT5MUX0TO15CFG Register Field Descriptions .....	1081
9-46.	OUTPUT5MUX16TO31CFG Register Field Descriptions.....	1084
9-47.	OUTPUT6MUX0TO15CFG Register Field Descriptions .....	1087
9-48.	OUTPUT6MUX16TO31CFG Register Field Descriptions.....	1090
9-49.	OUTPUT7MUX0TO15CFG Register Field Descriptions .....	1093
9-50.	OUTPUT7MUX16TO31CFG Register Field Descriptions.....	1096
9-51.	OUTPUT8MUX0TO15CFG Register Field Descriptions .....	1099
9-52.	OUTPUT8MUX16TO31CFG Register Field Descriptions.....	1102
9-53.	OUTPUT1MUXENABLE Register Field Descriptions .....	1105
9-54.	OUTPUT2MUXENABLE Register Field Descriptions .....	1110
9-55.	OUTPUT3MUXENABLE Register Field Descriptions .....	1115
9-56.	OUTPUT4MUXENABLE Register Field Descriptions .....	1120
9-57.	OUTPUT5MUXENABLE Register Field Descriptions .....	1125
9-58.	OUTPUT6MUXENABLE Register Field Descriptions .....	1130
9-59.	OUTPUT7MUXENABLE Register Field Descriptions .....	1135
9-60.	OUTPUT8MUXENABLE Register Field Descriptions .....	1140
9-61.	OUTPUTLATCH Register Field Descriptions.....	1145
9-62.	OUTPUTLATCHCLR Register Field Descriptions .....	1147
9-63.	OUTPUTLATCHFRC Register Field Descriptions .....	1149
9-64.	OUTPUTLATCHENABLE Register Field Descriptions .....	1151
9-65.	OUTPUTINV Register Field Descriptions .....	1153
9-66.	OUTPUTLOCK Register Field Descriptions .....	1155
9-67.	EPWM_XBAR_REGS Registers .....	1156
9-68.	EPWM_XBAR_REGS Access Type Codes .....	1156
9-69.	TRIP4MUX0TO15CFG Register Field Descriptions .....	1158
9-70.	TRIP4MUX16TO31CFG Register Field Descriptions.....	1161
9-71.	TRIP5MUX0TO15CFG Register Field Descriptions .....	1164
9-72.	TRIP5MUX16TO31CFG Register Field Descriptions.....	1167
9-73.	TRIP7MUX0TO15CFG Register Field Descriptions .....	1170
9-74.	TRIP7MUX16TO31CFG Register Field Descriptions.....	1173
9-75.	TRIP8MUX0TO15CFG Register Field Descriptions .....	1176
9-76.	TRIP8MUX16TO31CFG Register Field Descriptions.....	1179
9-77.	TRIP9MUX0TO15CFG Register Field Descriptions .....	1182
9-78.	TRIP9MUX16TO31CFG Register Field Descriptions.....	1185
9-79.	TRIP10MUX0TO15CFG Register Field Descriptions.....	1188
9-80.	TRIP10MUX16TO31CFG Register Field Descriptions .....	1191
9-81.	TRIP11MUX0TO15CFG Register Field Descriptions.....	1194
9-82.	TRIP11MUX16TO31CFG Register Field Descriptions .....	1197
9-83.	TRIP12MUX0TO15CFG Register Field Descriptions.....	1200
9-84.	TRIP12MUX16TO31CFG Register Field Descriptions .....	1203
9-85.	TRIP4MUXENABLE Register Field Descriptions .....	1206
9-86.	TRIP5MUXENABLE Register Field Descriptions .....	1211
9-87.	TRIP7MUXENABLE Register Field Descriptions .....	1216
9-88.	TRIP8MUXENABLE Register Field Descriptions .....	1221
9-89.	TRIP9MUXENABLE Register Field Descriptions .....	1226

9-90.	TRIP10MUXENABLE Register Field Descriptions .....	1231
9-91.	TRIP11MUXENABLE Register Field Descriptions .....	1236
9-92.	TRIP12MUXENABLE Register Field Descriptions .....	1241
9-93.	TRIPOUTINV Register Field Descriptions .....	1246
9-94.	TRIPLOCK Register Field Descriptions .....	1248
9-95.	CLB_XBAR_REGS Registers .....	1249
9-96.	CLB_XBAR_REGS Access Type Codes.....	1249
9-97.	AUXSIG0MUX0TO15CFG Register Field Descriptions .....	1251
9-98.	AUXSIG0MUX16TO31CFG Register Field Descriptions .....	1255
9-99.	AUXSIG1MUX0TO15CFG Register Field Descriptions .....	1259
9-100.	AUXSIG1MUX16TO31CFG Register Field Descriptions .....	1263
9-101.	AUXSIG2MUX0TO15CFG Register Field Descriptions .....	1267
9-102.	AUXSIG2MUX16TO31CFG Register Field Descriptions .....	1271
9-103.	AUXSIG3MUX0TO15CFG Register Field Descriptions .....	1275
9-104.	AUXSIG3MUX16TO31CFG Register Field Descriptions .....	1279
9-105.	AUXSIG4MUX0TO15CFG Register Field Descriptions .....	1283
9-106.	AUXSIG4MUX16TO31CFG Register Field Descriptions .....	1287
9-107.	AUXSIG5MUX0TO15CFG Register Field Descriptions .....	1291
9-108.	AUXSIG5MUX16TO31CFG Register Field Descriptions .....	1295
9-109.	AUXSIG6MUX0TO15CFG Register Field Descriptions .....	1299
9-110.	AUXSIG6MUX16TO31CFG Register Field Descriptions .....	1303
9-111.	AUXSIG7MUX0TO15CFG Register Field Descriptions .....	1307
9-112.	AUXSIG7MUX16TO31CFG Register Field Descriptions .....	1311
9-113.	AUXSIG0MUXENABLE Register Field Descriptions .....	1315
9-114.	AUXSIG1MUXENABLE Register Field Descriptions .....	1320
9-115.	AUXSIG2MUXENABLE Register Field Descriptions .....	1325
9-116.	AUXSIG3MUXENABLE Register Field Descriptions .....	1330
9-117.	AUXSIG4MUXENABLE Register Field Descriptions .....	1335
9-118.	AUXSIG5MUXENABLE Register Field Descriptions .....	1340
9-119.	AUXSIG6MUXENABLE Register Field Descriptions .....	1345
9-120.	AUXSIG7MUXENABLE Register Field Descriptions .....	1350
9-121.	AUXSIGOUTINV Register Field Descriptions .....	1355
9-122.	AUXSIGLOCK Register Field Descriptions .....	1357
9-123.	EPWMXBAR Registers to Driverlib Functions.....	1358
9-124.	INPUTXBAR Registers to Driverlib Functions .....	1359
9-125.	OUTPUTXBAR Registers to Driverlib Functions .....	1360
9-126.	XBAR Registers to Driverlib Functions .....	1361
9-127.	CLBXBAR Registers to Driverlib Functions.....	1361
10-1.	DMA Trigger Source Options .....	1367
10-2.	BURSTSIZE vs DATASIZE Behavior.....	1371
10-3.	DMA Base Address Table.....	1378
10-4.	DMA_REGS Registers .....	1379
10-5.	DMA_REGS Access Type Codes.....	1379
10-6.	DMACTRL Register Field Descriptions.....	1380
10-7.	DEBUGCTRL Register Field Descriptions .....	1381
10-8.	PRIORITYCTRL1 Register Field Descriptions .....	1382
10-9.	PRIORITYSTAT Register Field Descriptions .....	1383
10-10.	DMA_CH_REGS Registers .....	1384
10-11.	DMA_CH_REGS Access Type Codes .....	1384



10-12. MODE Register Field Descriptions.....	1386
10-13. CONTROL Register Field Descriptions .....	1388
10-14. BURST_SIZE Register Field Descriptions .....	1390
10-15. BURST_COUNT Register Field Descriptions .....	1391
10-16. SRC_BURST_STEP Register Field Descriptions.....	1392
10-17. DST_BURST_STEP Register Field Descriptions .....	1393
10-18. TRANSFER_SIZE Register Field Descriptions.....	1394
10-19. TRANSFER_COUNT Register Field Descriptions .....	1395
10-20. SRC_TRANSFER_STEP Register Field Descriptions .....	1396
10-21. DST_TRANSFER_STEP Register Field Descriptions.....	1397
10-22. SRC_WRAP_SIZE Register Field Descriptions .....	1398
10-23. SRC_WRAP_COUNT Register Field Descriptions .....	1399
10-24. SRC_WRAP_STEP Register Field Descriptions .....	1400
10-25. DST_WRAP_SIZE Register Field Descriptions .....	1401
10-26. DST_WRAP_COUNT Register Field Descriptions.....	1402
10-27. DST_WRAP_STEP Register Field Descriptions .....	1403
10-28. SRC_BEG_ADDR_SHADOW Register Field Descriptions .....	1404
10-29. SRC_ADDR_SHADOW Register Field Descriptions .....	1405
10-30. SRC_BEG_ADDR_ACTIVE Register Field Descriptions .....	1406
10-31. SRC_ADDR_ACTIVE Register Field Descriptions.....	1407
10-32. DST_BEG_ADDR_SHADOW Register Field Descriptions .....	1408
10-33. DST_ADDR_SHADOW Register Field Descriptions .....	1409
10-34. DST_BEG_ADDR_ACTIVE Register Field Descriptions.....	1410
10-35. DST_ADDR_ACTIVE Register Field Descriptions .....	1411
10-36. DMA Registers to Driverlib Functions .....	1412
11-1. Event Selector Mux Signals.....	1418
11-2. ERAD Base Address Table .....	1422
11-3. ERAD_GLOBAL_REGS Registers.....	1423
11-4. ERAD_GLOBAL_REGS Access Type Codes .....	1423
11-5. GLBL_EVENT_STAT Register Field Descriptions .....	1424
11-6. GLBL_HALT_STAT Register Field Descriptions .....	1426
11-7. GLBL_ENABLE Register Field Descriptions.....	1428
11-8. GLBL_CTM_RESET Register Field Descriptions .....	1430
11-9. GLBL_OWNER Register Field Descriptions .....	1431
11-10. ERAD_HWBP_REGS Registers .....	1432
11-11. ERAD_HWBP_REGS Access Type Codes.....	1432
11-12. HWBP_MASK Register Field Descriptions .....	1433
11-13. HWBP_REF Register Field Descriptions.....	1434
11-14. HWBP_CLEAR Register Field Descriptions .....	1435
11-15. HWBP_CNTL Register Field Descriptions .....	1436
11-16. HWBP_STATUS Register Field Descriptions .....	1438
11-17. ERAD_COUNTER_REGS Registers .....	1439
11-18. ERAD_COUNTER_REGS Access Type Codes .....	1439
11-19. CTM_CNTL Register Field Descriptions .....	1440
11-20. CTM_STATUS Register Field Descriptions.....	1442
11-21. CTM_REF Register Field Descriptions .....	1443
11-22. CTM_COUNT Register Field Descriptions.....	1444
11-23. CTM_MAX_COUNT Register Field Descriptions .....	1445
11-24. CTM_INPUT_SEL Register Field Descriptions.....	1446



11-25. CTM_CLEAR Register Field Descriptions .....	1447
12-1. Analog Pins and Internal Connections .....	1454
12-2. Analog Signal Descriptions .....	1456
12-3. Reference Summary .....	1456
12-4. Analog Subsystem Base Address Table .....	1458
12-5. ANALOG_SUBSYS_REGS Registers.....	1459
12-6. ANALOG_SUBSYS_REGS Access Type Codes .....	1459
12-7. ANAREFPP Register Field Descriptions .....	1461
12-8. TSNSCTL Register Field Descriptions .....	1462
12-9. ANAREFCTL Register Field Descriptions.....	1463
12-10. VMONCTL Register Field Descriptions .....	1465
12-11. DCDCCTL Register Field Descriptions.....	1466
12-12. DCDCSTS Register Field Descriptions.....	1467
12-13. CMPHPMXSEL Register Field Descriptions.....	1468
12-14. CMPLPMXSEL Register Field Descriptions .....	1470
12-15. CMPPHNMXSEL Register Field Descriptions.....	1472
12-16. CMPLNMXSEL Register Field Descriptions .....	1473
12-17. LOCK Register Field Descriptions .....	1474
13-1. ADC Options and Configuration Levels .....	1478
13-2. Analog to 12-bit Digital Formulas .....	1480
13-3. 12-Bit Digital-to-Analog Formulas .....	1480
13-4. Channel Selection of Input Pins .....	1483
13-5. Example Requirements for Multiple Signal Sampling .....	1485
13-6. Example Connections for Multiple Signal Sampling .....	1485
13-7. DETECTCFG Settings.....	1496
13-8. ADC Timing Parameters .....	1500
13-9. ADC Timings in 12-bit Mode (SYSCLK Cycles).....	1501
13-10. ADC Base Address Table .....	1509
13-11. ADC_REGS Registers.....	1510
13-12. ADC_REGS Access Type Codes .....	1511
13-13. ADCCTL1 Register Field Descriptions .....	1513
13-14. ADCCTL2 Register Field Descriptions .....	1515
13-15. ADCBURSTCTL Register Field Descriptions.....	1516
13-16. ADCINTFLG Register Field Descriptions .....	1518
13-17. ADCINTFLGCLR Register Field Descriptions .....	1520
13-18. ADCINTOVF Register Field Descriptions .....	1522
13-19. ADCINTOVFCLR Register Field Descriptions.....	1524
13-20. ADCINTSEL1N2 Register Field Descriptions.....	1525
13-21. ADCINTSEL3N4 Register Field Descriptions.....	1527
13-22. ADCSOCPRCTL Register Field Descriptions .....	1529
13-23. ADCINTSOCSEL1 Register Field Descriptions .....	1532
13-24. ADCINTSOCSEL2 Register Field Descriptions .....	1534
13-25. ADCSOCFLG1 Register Field Descriptions .....	1536
13-26. ADCSOCFRC1 Register Field Descriptions .....	1541
13-27. ADCSOCOVF1 Register Field Descriptions .....	1546
13-28. ADCSOCOVFCLR1 Register Field Descriptions.....	1550
13-29. ADCSOC0CTL Register Field Descriptions .....	1554
13-30. ADCSOC1CTL Register Field Descriptions .....	1556
13-31. ADCSOC2CTL Register Field Descriptions .....	1558

13-32. ADCSOC3CTL Register Field Descriptions .....	1560
13-33. ADCSOC4CTL Register Field Descriptions .....	1562
13-34. ADCSOC5CTL Register Field Descriptions .....	1564
13-35. ADCSOC6CTL Register Field Descriptions .....	1566
13-36. ADCSOC7CTL Register Field Descriptions .....	1568
13-37. ADCSOC8CTL Register Field Descriptions .....	1570
13-38. ADCSOC9CTL Register Field Descriptions .....	1572
13-39. ADCSOC10CTL Register Field Descriptions .....	1574
13-40. ADCSOC11CTL Register Field Descriptions .....	1576
13-41. ADCSOC12CTL Register Field Descriptions .....	1578
13-42. ADCSOC13CTL Register Field Descriptions .....	1580
13-43. ADCSOC14CTL Register Field Descriptions .....	1582
13-44. ADCSOC15CTL Register Field Descriptions .....	1584
13-45. ADCEVTSTAT Register Field Descriptions.....	1586
13-46. ADCEVTCLR Register Field Descriptions .....	1588
13-47. ADCEVTSEL Register Field Descriptions.....	1590
13-48. ADCEVTINTSEL Register Field Descriptions .....	1592
13-49. ADCOSDETECT Register Field Descriptions .....	1594
13-50. ADCCOUNTER Register Field Descriptions.....	1595
13-51. ADCREV Register Field Descriptions .....	1596
13-52. ADCOFFTRIM Register Field Descriptions .....	1597
13-53. ADCPPB1CONFIG Register Field Descriptions.....	1598
13-54. ADCPPB1STAMP Register Field Descriptions.....	1600
13-55. ADCPPB1OFFCAL Register Field Descriptions .....	1601
13-56. ADCPPB1OFFREF Register Field Descriptions .....	1602
13-57. ADCPPB1TRIPHI Register Field Descriptions .....	1603
13-58. ADCPPB1TRIPL0 Register Field Descriptions .....	1604
13-59. ADCPPB2CONFIG Register Field Descriptions.....	1605
13-60. ADCPPB2STAMP Register Field Descriptions.....	1607
13-61. ADCPPB2OFFCAL Register Field Descriptions .....	1608
13-62. ADCPPB2OFFREF Register Field Descriptions .....	1609
13-63. ADCPPB2TRIPHI Register Field Descriptions .....	1610
13-64. ADCPPB2TRIPL0 Register Field Descriptions .....	1611
13-65. ADCPPB3CONFIG Register Field Descriptions.....	1612
13-66. ADCPPB3STAMP Register Field Descriptions.....	1614
13-67. ADCPPB3OFFCAL Register Field Descriptions .....	1615
13-68. ADCPPB3OFFREF Register Field Descriptions .....	1616
13-69. ADCPPB3TRIPHI Register Field Descriptions .....	1617
13-70. ADCPPB3TRIPL0 Register Field Descriptions .....	1618
13-71. ADCPPB4CONFIG Register Field Descriptions.....	1619
13-72. ADCPPB4STAMP Register Field Descriptions.....	1621
13-73. ADCPPB4OFFCAL Register Field Descriptions .....	1622
13-74. ADCPPB4OFFREF Register Field Descriptions .....	1623
13-75. ADCPPB4TRIPHI Register Field Descriptions .....	1624
13-76. ADCPPB4TRIPL0 Register Field Descriptions .....	1625
13-77. ADCINTCYCLE Register Field Descriptions.....	1626
13-78. ADCINLTRIM1 Register Field Descriptions.....	1627
13-79. ADCINLTRIM2 Register Field Descriptions.....	1628
13-80. ADCINLTRIM3 Register Field Descriptions.....	1629

13-81. ADC_RESULT_REGS Registers.....	1631
13-82. ADC_RESULT_REGS Access Type Codes.....	1631
13-83. ADCRESULT0 Register Field Descriptions.....	1632
13-84. ADCRESULT1 Register Field Descriptions.....	1633
13-85. ADCRESULT2 Register Field Descriptions.....	1634
13-86. ADCRESULT3 Register Field Descriptions.....	1635
13-87. ADCRESULT4 Register Field Descriptions.....	1636
13-88. ADCRESULT5 Register Field Descriptions.....	1637
13-89. ADCRESULT6 Register Field Descriptions.....	1638
13-90. ADCRESULT7 Register Field Descriptions.....	1639
13-91. ADCRESULT8 Register Field Descriptions.....	1640
13-92. ADCRESULT9 Register Field Descriptions.....	1641
13-93. ADCRESULT10 Register Field Descriptions.....	1642
13-94. ADCRESULT11 Register Field Descriptions.....	1643
13-95. ADCRESULT12 Register Field Descriptions.....	1644
13-96. ADCRESULT13 Register Field Descriptions.....	1645
13-97. ADCRESULT14 Register Field Descriptions.....	1646
13-98. ADCRESULT15 Register Field Descriptions.....	1647
13-99. ADCPPB1RESULT Register Field Descriptions.....	1648
13-100. ADCPPB2RESULT Register Field Descriptions.....	1649
13-101. ADCPPB3RESULT Register Field Descriptions.....	1650
13-102. ADCPPB4RESULT Register Field Descriptions.....	1651
13-103. ADC Registers to Driverlib Functions.....	1652
14-1. Ideal Gain Resistor Values.....	1658
14-2. Minimum Filter Resistance.....	1658
14-3. I2C Base Address Table.....	1665
14-4. PGA_REGS Registers.....	1666
14-5. PGA_REGS Access Type Codes.....	1666
14-6. PGACTL Register Field Descriptions.....	1667
14-7. PGALOCK Register Field Descriptions.....	1668
14-8. PGAGAIN3TRIM Register Field Descriptions.....	1669
14-9. PGAGAIN6TRIM Register Field Descriptions.....	1670
14-10. PGAGAIN12TRIM Register Field Descriptions.....	1671
14-11. PGAGAIN24TRIM Register Field Descriptions.....	1672
14-12. PGATYPE Register Field Descriptions.....	1673
15-1. DAC Supported Gain Mode Combinations.....	1676
15-2. DAC Base Address Table.....	1678
15-3. DAC_REGS Registers.....	1679
15-4. DAC_REGS Access Type Codes.....	1679
15-5. DACREV Register Field Descriptions.....	1680
15-6. DACCTL Register Field Descriptions.....	1681
15-7. DACVALA Register Field Descriptions.....	1682
15-8. DACVALS Register Field Descriptions.....	1683
15-9. DACOUTEN Register Field Descriptions.....	1684
15-10. DACLOCK Register Field Descriptions.....	1685
15-11. DACTRIM Register Field Descriptions.....	1686
15-12. DAC Registers to Driverlib Functions.....	1687
16-1. CMPSS Base Address Table.....	1697
16-2. CMPSS_REGS Registers.....	1698

16-3.	CMPSS_REGS Access Type Codes .....	1698
16-4.	COMPCTL Register Field Descriptions .....	1700
16-5.	COMPHYCTL Register Field Descriptions .....	1702
16-6.	COMPSTS Register Field Descriptions .....	1703
16-7.	COMPSTSCLR Register Field Descriptions .....	1704
16-8.	COMPDACCTL Register Field Descriptions .....	1705
16-9.	DACHVALS Register Field Descriptions .....	1707
16-10.	DACHVALA Register Field Descriptions .....	1708
16-11.	RAMPMAXREFA Register Field Descriptions .....	1709
16-12.	RAMPMAXREFS Register Field Descriptions .....	1710
16-13.	RAMPDECVALA Register Field Descriptions .....	1711
16-14.	RAMPDECVALS Register Field Descriptions .....	1712
16-15.	RAMPSTS Register Field Descriptions .....	1713
16-16.	DACLVALS Register Field Descriptions .....	1714
16-17.	DACLVALA Register Field Descriptions .....	1715
16-18.	RAMPDLYA Register Field Descriptions .....	1716
16-19.	RAMPDLYS Register Field Descriptions .....	1717
16-20.	CTRIPLFILCTL Register Field Descriptions .....	1718
16-21.	CTRIPLFILCLKCTL Register Field Descriptions .....	1719
16-22.	CTRIPHFILCTL Register Field Descriptions .....	1720
16-23.	CTRIPHFILCLKCTL Register Field Descriptions .....	1721
16-24.	COMPLOCK Register Field Descriptions .....	1722
16-25.	CMPSS Registers to Driverlib Functions .....	1723
17-1.	Modulator Clock Modes .....	1730
17-2.	Order of Sinc Filter .....	1733
17-3.	Peak Data Values for Different DOSR/Filter Combinations .....	1734
17-4.	Shift Control Bit Configuration Settings .....	1735
17-5.	SDSYNCx.SYNCSEL .....	1736
17-6.	Number of Incorrect Samples Tabulated .....	1737
17-7.	Peak Data Values for Different OSR/Filter Combinations .....	1738
17-8.	<b>SDFM Data-Ready Interrupt (SDy_DRINTx) Output Selection</b> .....	1742
17-9.	SDFM Base Address Table .....	1743
17-10.	SDFM_REGS Registers .....	1744
17-11.	SDFM_REGS Access Type Codes .....	1745
17-12.	SDIFLG Register Field Descriptions .....	1746
17-13.	SDIFLGCLR Register Field Descriptions .....	1749
17-14.	SDCTL Register Field Descriptions .....	1751
17-15.	SDMFILEN Register Field Descriptions .....	1752
17-16.	SDSTATUS Register Field Descriptions .....	1753
17-17.	SDCTLPARM1 Register Field Descriptions .....	1755
17-18.	SDDFPARM1 Register Field Descriptions .....	1756
17-19.	SDDPARM1 Register Field Descriptions .....	1757
17-20.	SDCMPH1 Register Field Descriptions .....	1758
17-21.	SDCMPL1 Register Field Descriptions .....	1759
17-22.	SDCPARM1 Register Field Descriptions .....	1760
17-23.	SDDATA1 Register Field Descriptions .....	1762
17-24.	SDDATFIFO1 Register Field Descriptions .....	1763
17-25.	SDCDATA1 Register Field Descriptions .....	1764
17-26.	SDCMPHZ1 Register Field Descriptions .....	1765

17-27. SDFIFOCTL1 Register Field Descriptions .....	1766
17-28. SDSYNC1 Register Field Descriptions .....	1767
17-29. SDCTLPARM2 Register Field Descriptions .....	1768
17-30. SDDFPARM2 Register Field Descriptions .....	1769
17-31. SDDPARM2 Register Field Descriptions.....	1770
17-32. SDCMPH2 Register Field Descriptions .....	1771
17-33. SDCMPL2 Register Field Descriptions .....	1772
17-34. SDCPARM2 Register Field Descriptions.....	1773
17-35. SDDATA2 Register Field Descriptions .....	1775
17-36. SDDATFIFO2 Register Field Descriptions.....	1776
17-37. SDCDATA2 Register Field Descriptions .....	1777
17-38. SDCMPHZ2 Register Field Descriptions.....	1778
17-39. SDFIFOCTL2 Register Field Descriptions .....	1779
17-40. SDSYNC2 Register Field Descriptions .....	1780
17-41. SDCTLPARM3 Register Field Descriptions .....	1781
17-42. SDDFPARM3 Register Field Descriptions .....	1782
17-43. SDDPARM3 Register Field Descriptions.....	1783
17-44. SDCMPH3 Register Field Descriptions .....	1784
17-45. SDCMPL3 Register Field Descriptions .....	1785
17-46. SDCPARM3 Register Field Descriptions.....	1786
17-47. SDDATA3 Register Field Descriptions .....	1788
17-48. SDDATFIFO3 Register Field Descriptions.....	1789
17-49. SDCDATA3 Register Field Descriptions .....	1790
17-50. SDCMPHZ3 Register Field Descriptions.....	1791
17-51. SDFIFOCTL3 Register Field Descriptions .....	1792
17-52. SDSYNC3 Register Field Descriptions .....	1793
17-53. SDCTLPARM4 Register Field Descriptions .....	1794
17-54. SDDFPARM4 Register Field Descriptions .....	1795
17-55. SDDPARM4 Register Field Descriptions.....	1796
17-56. SDCMPH4 Register Field Descriptions .....	1797
17-57. SDCMPL4 Register Field Descriptions .....	1798
17-58. SDCPARM4 Register Field Descriptions.....	1799
17-59. SDDATA4 Register Field Descriptions .....	1801
17-60. SDDATFIFO4 Register Field Descriptions.....	1802
17-61. SDCDATA4 Register Field Descriptions .....	1803
17-62. SDCMPHZ4 Register Field Descriptions.....	1804
17-63. SDFIFOCTL4 Register Field Descriptions .....	1805
17-64. SDSYNC4 Register Field Descriptions .....	1806
17-65. SDFM Registers to Driverlib Functions.....	1807
18-1. Submodule Configuration Parameters.....	1818
18-2. Key Time-Base Signals.....	1821
18-3. Action-Qualifier Submodule Possible Input Events .....	1839
18-4. Action-Qualifier Event Priority for Up-Down-Count Mode.....	1841
18-5. Action-Qualifier Event Priority for Up-Count Mode.....	1841
18-6. Action-Qualifier Event Priority for Down-Count Mode .....	1841
18-7. Behavior if CMPA/CMPB is Greater than the Period.....	1842
18-8. Classical Dead-Band Operating Modes .....	1854
18-9. Additional Dead-Band Operating Modes .....	1854
18-10. Dead-Band Delay Values in $\mu$ S as a Function of DBFED and DBRED .....	1856

18-11. Possible Pulse Width Values for EPWMCLK = 80 MHz .....	1859
18-12. Possible Actions On a Trip Event .....	1863
18-13. ePWM X-BAR Mux Configuration Table .....	1883
18-14. Lock Bits and Corresponding Registers .....	1902
18-15. Resolution for PWM and HRPWM .....	1904
18-16. Relationship Between MEP Steps, PWM Frequency and Resolution .....	1911
18-17. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right).....	1912
18-18. Duty Cycle Range Limitation for Three EPWMCLK/TBCLK Cycles .....	1915
18-19. SFO Library Features.....	1927
18-20. Factor Values .....	1928
18-21. ePWM Base Address Table .....	1930
18-22. EPWM_REGS Registers .....	1931
18-23. EPWM_REGS Access Type Codes .....	1933
18-24. TBCTL Register Field Descriptions .....	1934
18-25. TBCTL2 Register Field Descriptions.....	1937
18-26. TBCTR Register Field Descriptions.....	1938
18-27. TBSTS Register Field Descriptions .....	1939
18-28. CMPCTL Register Field Descriptions.....	1940
18-29. CMPCTL2 Register Field Descriptions .....	1942
18-30. DBCTL Register Field Descriptions .....	1944
18-31. DBCTL2 Register Field Descriptions .....	1947
18-32. AQCTL Register Field Descriptions.....	1948
18-33. AQTSRCSEL Register Field Descriptions .....	1950
18-34. PCCTL Register Field Descriptions .....	1951
18-35. VCAPCTL Register Field Descriptions .....	1953
18-36. VCNTCFG Register Field Descriptions.....	1955
18-37. HRCNFG Register Field Descriptions .....	1957
18-38. HRPWR Register Field Descriptions .....	1959
18-39. HRMSTEP Register Field Descriptions .....	1960
18-40. HRCNFG2 Register Field Descriptions.....	1961
18-41. HRPCTL Register Field Descriptions .....	1962
18-42. TRREM Register Field Descriptions .....	1964
18-43. GLDCTL Register Field Descriptions .....	1965
18-44. GLDCFG Register Field Descriptions .....	1967
18-45. EPWMLINK Register Field Descriptions .....	1969
18-46. AQCTLA Register Field Descriptions.....	1973
18-47. AQCTLA2 Register Field Descriptions .....	1975
18-48. AQCTLB Register Field Descriptions .....	1977
18-49. AQCTLB2 Register Field Descriptions .....	1979
18-50. AQSFRG Register Field Descriptions .....	1981
18-51. AQCSFRG Register Field Descriptions .....	1983
18-52. DBREDHR Register Field Descriptions .....	1984
18-53. DBRED Register Field Descriptions .....	1985
18-54. DBFEDHR Register Field Descriptions.....	1986
18-55. DBFED Register Field Descriptions .....	1987
18-56. TBPFS Register Field Descriptions.....	1988
18-57. TBPRDHR Register Field Descriptions.....	1989
18-58. TBPRD Register Field Descriptions .....	1990
18-59. CMPA Register Field Descriptions .....	1991



18-60. CMPB Register Field Descriptions .....	1992
18-61. CMPC Register Field Descriptions .....	1993
18-62. CMPD Register Field Descriptions .....	1994
18-63. GLDCTL2 Register Field Descriptions .....	1995
18-64. SWVDELVAL Register Field Descriptions .....	1996
18-65. TZSEL Register Field Descriptions .....	1997
18-66. TZDCSEL Register Field Descriptions .....	1999
18-67. TZCTL Register Field Descriptions .....	2001
18-68. TZCTL2 Register Field Descriptions .....	2003
18-69. TZCTLDCA Register Field Descriptions .....	2005
18-70. TZCTLDCB Register Field Descriptions .....	2007
18-71. TZEINT Register Field Descriptions .....	2009
18-72. TZFLG Register Field Descriptions .....	2010
18-73. TZCBCFLG Register Field Descriptions .....	2012
18-74. TZOSTFLG Register Field Descriptions .....	2014
18-75. TZCLR Register Field Descriptions .....	2016
18-76. TZCBCCLR Register Field Descriptions .....	2018
18-77. TZOSTCLR Register Field Descriptions .....	2020
18-78. TZFRC Register Field Descriptions .....	2022
18-79. ETSEL Register Field Descriptions .....	2023
18-80. ETPS Register Field Descriptions .....	2026
18-81. ETFLG Register Field Descriptions .....	2029
18-82. ETCLR Register Field Descriptions .....	2030
18-83. ETFRC Register Field Descriptions .....	2031
18-84. ETINTPS Register Field Descriptions .....	2032
18-85. ETSOCPS Register Field Descriptions .....	2033
18-86. ETCNTINITCTL Register Field Descriptions .....	2035
18-87. ETCNTINIT Register Field Descriptions .....	2036
18-88. DCTRIPSEL Register Field Descriptions .....	2037
18-89. DCACTL Register Field Descriptions .....	2039
18-90. DCBCTL Register Field Descriptions .....	2041
18-91. DCFCTL Register Field Descriptions .....	2043
18-92. DCCAPCTL Register Field Descriptions .....	2045
18-93. DCFOFFSET Register Field Descriptions .....	2047
18-94. DCFOFFSETCNT Register Field Descriptions .....	2048
18-95. DCFWINDOW Register Field Descriptions .....	2049
18-96. DCFWINDOWCNT Register Field Descriptions .....	2050
18-97. DCCAP Register Field Descriptions .....	2051
18-98. DCAHTRIPSEL Register Field Descriptions .....	2052
18-99. DCALTRIPSEL Register Field Descriptions .....	2054
18-100. DCBHTRIPSEL Register Field Descriptions .....	2056
18-101. DCBLTRIPSEL Register Field Descriptions .....	2058
18-102. EPWMLOCK Register Field Descriptions .....	2060
18-103. HWVDELVAL Register Field Descriptions .....	2062
18-104. VCNTVAL Register Field Descriptions .....	2063
18-105. SYNC_SOC_REGS Registers .....	2064
18-106. SYNC_SOC_REGS Access Type Codes .....	2064
18-107. SYNCSELECT Register Field Descriptions .....	2065
18-108. ADCSOCOUTSELECT Register Field Descriptions .....	2068



18-109. SYNCLOCK Register Field Descriptions.....	2071
18-110. EPWM Registers to Driverlib Functions.....	2072
18-111. HRPWM Registers to Driverlib Functions .....	2078
19-1. eCAP Input Selection .....	2082
19-2. eCAP Base Address Table .....	2100
19-3. ECAP_REGS Registers .....	2101
19-4. ECAP_REGS Access Type Codes .....	2101
19-5. TSCTR Register Field Descriptions.....	2102
19-6. CTRPHS Register Field Descriptions.....	2103
19-7. CAP1 Register Field Descriptions.....	2104
19-8. CAP2 Register Field Descriptions.....	2105
19-9. CAP3 Register Field Descriptions.....	2106
19-10. CAP4 Register Field Descriptions.....	2107
19-11. ECCTL0 Register Field Descriptions .....	2108
19-12. ECCTL1 Register Field Descriptions .....	2109
19-13. ECCTL2 Register Field Descriptions .....	2111
19-14. ECEINT Register Field Descriptions.....	2114
19-15. ECFLG Register Field Descriptions.....	2116
19-16. ECCLR Register Field Descriptions .....	2118
19-17. ECFRC Register Field Descriptions .....	2119
19-18. ECAP Registers to Driverlib Functions .....	2120
20-1. Scale Factor.....	2127
20-2. HRCAP Base Address Table .....	2128
20-3. HRCAP_REGS Registers .....	2129
20-4. HRCAP_REGS Access Type Codes .....	2129
20-5. HRCTL Register Field Descriptions.....	2130
20-6. HRINTEN Register Field Descriptions.....	2132
20-7. HRFLG Register Field Descriptions .....	2133
20-8. HRCLR Register Field Descriptions .....	2134
20-9. HRFRC Register Field Descriptions .....	2135
20-10. HRCALPRD Register Field Descriptions .....	2136
20-11. HRSYSCLKCTR Register Field Descriptions.....	2137
20-12. HRSYSCLKCAP Register Field Descriptions.....	2138
20-13. HRCLKCTR Register Field Descriptions .....	2139
20-14. HRCLKCAP Register Field Descriptions .....	2140
21-1. EQEP Memory Map .....	2146
21-2. Quadrature Decoder Truth Table .....	2148
21-3. eQEP Base Address Table .....	2167
21-4. EQEP_REGS Registers .....	2168
21-5. EQEP_REGS Access Type Codes .....	2168
21-6. QPOSCNT Register Field Descriptions .....	2170
21-7. QPOSINIT Register Field Descriptions.....	2171
21-8. QPOSMAX Register Field Descriptions .....	2172
21-9. QPOSCMP Register Field Descriptions.....	2173
21-10. QPOSILAT Register Field Descriptions .....	2174
21-11. QPOSSLAT Register Field Descriptions .....	2175
21-12. QPOSLAT Register Field Descriptions .....	2176
21-13. QUTMR Register Field Descriptions.....	2177
21-14. QUPRD Register Field Descriptions .....	2178

21-15. QWDTMR Register Field Descriptions .....	2179
21-16. QWDPRD Register Field Descriptions .....	2180
21-17. QDECCTL Register Field Descriptions .....	2181
21-18. QEPCTL Register Field Descriptions .....	2183
21-19. QCAPCTL Register Field Descriptions .....	2186
21-20. QPOSCTL Register Field Descriptions .....	2187
21-21. QEINT Register Field Descriptions .....	2188
21-22. QFLG Register Field Descriptions .....	2190
21-23. QCLR Register Field Descriptions .....	2192
21-24. QFRC Register Field Descriptions .....	2194
21-25. QEPSTS Register Field Descriptions .....	2196
21-26. QCTMR Register Field Descriptions .....	2198
21-27. QCPRD Register Field Descriptions .....	2199
21-28. QCTMRLAT Register Field Descriptions .....	2200
21-29. QCPRDLAT Register Field Descriptions .....	2201
21-30. REV Register Field Descriptions .....	2202
21-31. QEPSTROBESEL Register Field Descriptions .....	2203
21-32. QMACTRL Register Field Descriptions .....	2204
21-33. EQEP Registers to Driverlib Functions .....	2205
22-1. SPI Module Signal Summary .....	2209
22-2. SPI Interrupt Flag Modes .....	2212
22-3. SPI Clocking Scheme Selection Guide .....	2217
22-4. 4-wire vs. 3-wire SPI Pin Functions .....	2220
22-5. 3-Wire SPI Pin Configuration .....	2221
22-6. SPI Base Address Table .....	2227
22-7. SPI_REGS Registers .....	2228
22-8. SPI_REGS Access Type Codes .....	2228
22-9. SPICCR Register Field Descriptions .....	2229
22-10. SPICTL Register Field Descriptions .....	2231
22-11. SPISTS Register Field Descriptions .....	2233
22-12. SPIBRR Register Field Descriptions .....	2235
22-13. SPIRXEMU Register Field Descriptions .....	2236
22-14. SPIRXBUF Register Field Descriptions .....	2237
22-15. SPITXBUF Register Field Descriptions .....	2238
22-16. SPIDAT Register Field Descriptions .....	2239
22-17. SPIFFTX Register Field Descriptions .....	2240
22-18. SPIFFRX Register Field Descriptions .....	2242
22-19. SPIFFCT Register Field Descriptions .....	2244
22-20. SPIPRI Register Field Descriptions .....	2245
22-21. SPI Registers to Driverlib Functions .....	2247
23-1. SCI Module Signal Summary .....	2252
23-2. Programming the Data Format Using SCICCR .....	2253
23-3. Asynchronous Baud Register Values for Common SCI Bit Rates .....	2260
23-4. SCI Interrupt Flags .....	2262
23-5. SCI Base Address Table .....	2263
23-6. SCI_REGS Registers .....	2264
23-7. SCI_REGS Access Type Codes .....	2264
23-8. SCICCR Register Field Descriptions .....	2265
23-9. SCICTL1 Register Field Descriptions .....	2267

23-10. SCIHBAUD Register Field Descriptions.....	2269
23-11. SCILBAUD Register Field Descriptions .....	2270
23-12. SCICTL2 Register Field Descriptions.....	2271
23-13. SCIRXST Register Field Descriptions .....	2273
23-14. SCIRXEMU Register Field Descriptions .....	2275
23-15. SCIRXBUF Register Field Descriptions .....	2276
23-16. SCITXBUF Register Field Descriptions .....	2277
23-17. SCIFFTX Register Field Descriptions .....	2278
23-18. SCIFFRX Register Field Descriptions .....	2280
23-19. SCIFFCT Register Field Descriptions .....	2282
23-20. SCIPRI Register Field Descriptions.....	2283
23-21. SCI Registers to Driverlib Functions .....	2284
24-1. Dependency of Delay d on the Divide-Down Value IPSC .....	2290
24-2. Operating Modes of the I2C Module.....	2291
24-3. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR .....	2292
24-4. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR .....	2295
24-5. Ways to Generate a NACK Bit .....	2296
24-6. Descriptions of the Basic I2C Interrupt Requests .....	2299
24-7. I2C Base Address Table .....	2302
24-8. I2C_REGS Registers .....	2303
24-9. I2C_REGS Access Type Codes.....	2303
24-10. I2COAR Register Field Descriptions.....	2304
24-11. I2CIER Register Field Descriptions .....	2305
24-12. I2CSTR Register Field Descriptions .....	2306
24-13. I2CCLKL Register Field Descriptions.....	2310
24-14. I2CCLKH Register Field Descriptions .....	2311
24-15. I2CCNT Register Field Descriptions .....	2312
24-16. I2CDRR Register Field Descriptions.....	2313
24-17. I2CSAR Register Field Descriptions .....	2314
24-18. I2CDXR Register Field Descriptions.....	2315
24-19. I2CMDR Register Field Descriptions .....	2316
24-20. I2CISRC Register Field Descriptions .....	2320
24-21. I2CEMDR Register Field Descriptions .....	2321
24-22. I2CPSC Register Field Descriptions .....	2322
24-23. I2CFFTX Register Field Descriptions.....	2323
24-24. I2CFFRX Register Field Descriptions .....	2325
24-25. I2C Registers to Driverlib Functions .....	2327
25-1. PMBus Base Address Table .....	2349
25-2. PMBUS_REGS Registers .....	2350
25-3. PMBUS_REGS Access Type Codes .....	2350
25-4. PMBMC Register Field Descriptions.....	2351
25-5. PMBTXBUF Register Field Descriptions .....	2353
25-6. PMBRXBUF Register Field Descriptions.....	2354
25-7. PMBACK Register Field Descriptions .....	2355
25-8. PMBSTS Register Field Descriptions.....	2356
25-9. PMBINTM Register Field Descriptions .....	2358
25-10. PMBSC Register Field Descriptions .....	2360
25-11. PMBHSA Register Field Descriptions .....	2362
25-12. PMBCTRL Register Field Descriptions.....	2363

25-13. PMBTIMCTL Register Field Descriptions .....	2366
25-14. PMBTIMCLK Register Field Descriptions .....	2367
25-15. PMBTIMSTSETUP Register Field Descriptions.....	2368
25-16. PMBTIMBIDLE Register Field Descriptions .....	2369
25-17. PMBTIMLOWTIMOUT Register Field Descriptions.....	2370
25-18. PMBTIMHIGHTIMOUT Register Field Descriptions .....	2371
26-1. CAN Register Access From Software .....	2376
26-2. CAN Register Access From CCS .....	2378
26-3. PIE Nomenclature for Interrupts.....	2384
26-4. Programmable Ranges Required by CAN Protocol .....	2394
26-5. Message Object Field Descriptions .....	2404
26-6. Message RAM Addressing in Debug Mode .....	2406
26-7. CAN Base Address Table .....	2408
26-8. CAN_REGS Registers.....	2409
26-9. CAN_REGS Access Type Codes .....	2410
26-10. CAN_CTL Register Field Descriptions .....	2411
26-11. CAN_ES Register Field Descriptions .....	2414
26-12. CAN_ERRC Register Field Descriptions .....	2416
26-13. CAN_BTR Register Field Descriptions .....	2417
26-14. CAN_INT Register Field Descriptions .....	2419
26-15. CAN_TEST Register Field Descriptions.....	2420
26-16. CAN_PERR Register Field Descriptions .....	2422
26-17. CAN_RAM_INIT Register Field Descriptions .....	2423
26-18. CAN_GLB_INT_EN Register Field Descriptions .....	2424
26-19. CAN_GLB_INT_FLG Register Field Descriptions .....	2425
26-20. CAN_GLB_INT_CLR Register Field Descriptions .....	2426
26-21. CAN_ABOTR Register Field Descriptions .....	2427
26-22. CAN_TXRQ_X Register Field Descriptions.....	2428
26-23. CAN_TXRQ_21 Register Field Descriptions .....	2429
26-24. CAN_NDAT_X Register Field Descriptions.....	2430
26-25. CAN_NDAT_21 Register Field Descriptions.....	2431
26-26. CAN_IPEN_X Register Field Descriptions .....	2432
26-27. CAN_IPEN_21 Register Field Descriptions.....	2433
26-28. CAN_MVAL_X Register Field Descriptions.....	2434
26-29. CAN_MVAL_21 Register Field Descriptions.....	2435
26-30. CAN_IP_MUX21 Register Field Descriptions .....	2436
26-31. CAN_IF1CMD Register Field Descriptions .....	2437
26-32. CAN_IF1MSK Register Field Descriptions.....	2441
26-33. CAN_IF1ARB Register Field Descriptions .....	2442
26-34. CAN_IF1MCTL Register Field Descriptions .....	2444
26-35. CAN_IF1DATA Register Field Descriptions .....	2447
26-36. CAN_IF1DATB Register Field Descriptions .....	2448
26-37. CAN_IF2CMD Register Field Descriptions .....	2449
26-38. CAN_IF2MSK Register Field Descriptions.....	2453
26-39. CAN_IF2ARB Register Field Descriptions .....	2454
26-40. CAN_IF2MCTL Register Field Descriptions .....	2456
26-41. CAN_IF2DATA Register Field Descriptions .....	2459
26-42. CAN_IF2DATB Register Field Descriptions .....	2460
26-43. CAN_IF3OBS Register Field Descriptions.....	2461

26-44. CAN_IF3MSK Register Field Descriptions .....	2463
26-45. CAN_IF3ARB Register Field Descriptions .....	2464
26-46. CAN_IF3MCTL Register Field Descriptions .....	2465
26-47. CAN_IF3DATA Register Field Descriptions .....	2467
26-48. CAN_IF3DATB Register Field Descriptions .....	2468
26-49. CAN_IF3UPD Register Field Descriptions .....	2469
26-50. CAN Registers to Driverlib Functions.....	2470
27-1. P and M values for standard bit-rates .....	2480
27-2. Superfractional Bit Modulation for SCI Mode (Normal Configuration) .....	2481
27-3. Superfractional Bit Modulation for SCI Mode (Maximum Configuration) .....	2481
27-4. SCI Mode (Minimum Configuration) .....	2481
27-5. Comparative Baud Values for Different P Values, Asynchronous Mode .....	2482
27-6. SCI/LIN Interrupts .....	2489
27-7. SCI Receiver Status Flags .....	2489
27-8. SCI Transmitter Status Flags .....	2490
27-9. Response Length Info Using IDBYTE Field Bits [5:4] for LIN Standards Earlier than 1.3.....	2497
27-10. Response Length with SCIFORMAT[18:16] Programming .....	2497
27-11. Superfractional Bit Modulation for LIN Master Mode and Slave Mode .....	2499
27-12. Timeout Values in T <sub>bit</sub> Units .....	2505
27-13. I2C Base Address Table .....	2519
27-14. LIN_REGS Registers .....	2520
27-15. LIN_REGS Access Type Codes.....	2520
27-16. SCIGCR0 Register Field Descriptions.....	2522
27-17. SCIGCR1 Register Field Descriptions.....	2523
27-18. SCIGCR2 Register Field Descriptions.....	2528
27-19. SCISSETINT Register Field Descriptions.....	2530
27-20. SCICLEARINT Register Field Descriptions.....	2534
27-21. SCISSETINTLVL Register Field Descriptions.....	2538
27-22. SCICLEARINTLVL Register Field Descriptions .....	2541
27-23. SCIFLR Register Field Descriptions .....	2545
27-24. SCIINTVECT0 Register Field Descriptions .....	2554
27-25. SCIINTVECT1 Register Field Descriptions .....	2555
27-26. SCIFORMAT Register Field Descriptions.....	2556
27-27. BRSR Register Field Descriptions .....	2558
27-28. SCIED Register Field Descriptions .....	2560
27-29. SCIRD Register Field Descriptions .....	2561
27-30. SCITD Register Field Descriptions.....	2562
27-31. SCIPIO0 Register Field Descriptions .....	2563
27-32. SCIPIO2 Register Field Descriptions .....	2564
27-33. LINCMP Register Field Descriptions .....	2565
27-34. LINRD0 Register Field Descriptions .....	2566
27-35. LINRD1 Register Field Descriptions .....	2567
27-36. LINMASK Register Field Descriptions.....	2568
27-37. LINID Register Field Descriptions.....	2569
27-38. LINTD0 Register Field Descriptions .....	2570
27-39. LINTD1 Register Field Descriptions .....	2571
27-40. MBRSR Register Field Descriptions.....	2572
27-41. IODFTCTRL Register Field Descriptions .....	2573
27-42. LIN_GLB_INT_EN Register Field Descriptions .....	2576

27-43. LIN_GLB_INT_FLG Register Field Descriptions .....	2577
27-44. LIN_GLB_INT_CLR Register Field Descriptions .....	2578
28-1. FSI Transmitter Core Signals .....	2582
28-2. FSI Receiver Core Signals .....	2583
28-3. External Trigger Sources and Their Index .....	2586
28-4. Basic Frame Structure .....	2598
28-5. Frame Types and their 4-bit codes .....	2600
28-6. Ping Frame .....	2600
28-7. Error Frame .....	2600
28-8. Data Frame .....	2601
28-9. Multi-Lane Frame Format .....	2601
28-10. SPI Compatibility Frame Structure .....	2604
28-11. Contents of Data Received by a Standard SPI .....	2604
28-12. FSI as Master Transmitter, SPI as Slave Receiver .....	2604
28-13. SPI as master transmitter, FSI as slave receiver .....	2606
28-14. FSI Base Address Table .....	2610
28-15. FSI_RX_REGS Registers .....	2611
28-16. FSI_RX_REGS Access Type Codes .....	2611
28-17. RX_MASTER_CTRL Register Field Descriptions.....	2613
28-18. RX_OPER_CTRL Register Field Descriptions .....	2615
28-19. RX_FRAME_INFO Register Field Descriptions .....	2617
28-20. RX_FRAME_TAG_UDATA Register Field Descriptions .....	2618
28-21. RX_DMA_CTRL Register Field Descriptions .....	2619
28-22. RX_EVT_STS Register Field Descriptions .....	2620
28-23. RX_CRC_INFO Register Field Descriptions.....	2623
28-24. RX_EVT_CLR Register Field Descriptions .....	2624
28-25. RX_EVT_FRC Register Field Descriptions .....	2626
28-26. RX_BUF_PTR_LOAD Register Field Descriptions .....	2629
28-27. RX_BUF_PTR_STS Register Field Descriptions .....	2630
28-28. RX_FRAME_WD_CTRL Register Field Descriptions .....	2631
28-29. RX_FRAME_WD_REF Register Field Descriptions .....	2632
28-30. RX_FRAME_WD_CNT Register Field Descriptions .....	2633
28-31. RX_PING_WD_CTRL Register Field Descriptions .....	2634
28-32. RX_PING_TAG Register Field Descriptions.....	2635
28-33. RX_PING_WD_REF Register Field Descriptions .....	2636
28-34. RX_PING_WD_CNT Register Field Descriptions.....	2637
28-35. RX_INT1_CTRL Register Field Descriptions .....	2638
28-36. RX_INT2_CTRL Register Field Descriptions .....	2641
28-37. RX_LOCK_CTRL Register Field Descriptions .....	2644
28-38. RX_ECC_DATA Register Field Descriptions .....	2645
28-39. RX_ECC_VAL Register Field Descriptions .....	2646
28-40. RX_ECC_SEC_DATA Register Field Descriptions .....	2647
28-41. RX_ECC_LOG Register Field Descriptions .....	2648
28-42. RX_DLYLINE_CTRL Register Field Descriptions.....	2649
28-43. RX_VIS_1 Register Field Descriptions .....	2650
28-44. RX_BUF_BASE_y Register Field Descriptions .....	2651
28-45. FSI_TX_REGS Registers .....	2652
28-46. FSI_TX_REGS Access Type Codes.....	2652
28-47. TX_MASTER_CTRL Register Field Descriptions .....	2654



28-48. TX_CLK_CTRL Register Field Descriptions .....	2655
28-49. TX_OPER_CTRL_LO Register Field Descriptions .....	2656
28-50. TX_OPER_CTRL_HI Register Field Descriptions .....	2658
28-51. TX_FRAME_CTRL Register Field Descriptions.....	2659
28-52. TX_FRAME_TAG_UDATA Register Field Descriptions.....	2661
28-53. TX_BUF_PTR_LOAD Register Field Descriptions.....	2662
28-54. TX_BUF_PTR_STS Register Field Descriptions.....	2663
28-55. TX_PING_CTRL Register Field Descriptions.....	2664
28-56. TX_PING_TAG Register Field Descriptions .....	2665
28-57. TX_PING_TO_REF Register Field Descriptions .....	2666
28-58. TX_PING_TO_CNT Register Field Descriptions .....	2667
28-59. TX_INT_CTRL Register Field Descriptions .....	2668
28-60. TX_DMA_CTRL Register Field Descriptions .....	2670
28-61. TX_LOCK_CTRL Register Field Descriptions.....	2671
28-62. TX_EVT_STS Register Field Descriptions.....	2672
28-63. TX_EVT_CLR Register Field Descriptions.....	2674
28-64. TX_EVT_FRC Register Field Descriptions .....	2675
28-65. TX_USER_CRC Register Field Descriptions.....	2676
28-66. TX_ECC_DATA Register Field Descriptions .....	2677
28-67. TX_ECC_VAL Register Field Descriptions .....	2678
28-68. TX_BUF_BASE_y Register Field Descriptions.....	2679
28-69. FSI Registers to Driverlib Functions .....	2680
29-1. Global Signals and Mux Selection .....	2687
29-2. Local Signals and Mux Selection.....	2689
29-3. Peripheral Signal Multiplexer Table.....	2691
29-4. Output Table .....	2694
29-5. Input Table .....	2695
29-6. Ports Tied Off to Prevent Combinatorial Loops .....	2695
29-7. Counter Block Operating Modes .....	2697
29-8. HLC Event List.....	2701
29-9. HLC Instruction Address Ranges .....	2701
29-10. Instruction Format .....	2702
29-11. HLC Register Encoding .....	2703
29-12. Non-Memory Mapped Register Addresses .....	2704
29-13. CLB Base Address Table (C28).....	2705
29-14. CLB_LOGIC_CONFIG_REGS Registers .....	2706
29-15. CLB_LOGIC_CONFIG_REGS Access Type Codes.....	2706
29-16. CLB_COUNT_RESET Register Field Descriptions.....	2708
29-17. CLB_COUNT_MODE_1 Register Field Descriptions.....	2709
29-18. CLB_COUNT_MODE_0 Register Field Descriptions.....	2710
29-19. CLB_COUNT_EVENT Register Field Descriptions.....	2711
29-20. CLB_FSM_EXTRA_IN0 Register Field Descriptions .....	2712
29-21. CLB_FSM_EXTERNAL_IN0 Register Field Descriptions.....	2713
29-22. CLB_FSM_EXTERNAL_IN1 Register Field Descriptions.....	2714
29-23. CLB_FSM_EXTRA_IN1 Register Field Descriptions .....	2715
29-24. CLB_LUT4_IN0 Register Field Descriptions.....	2716
29-25. CLB_LUT4_IN1 Register Field Descriptions.....	2717
29-26. CLB_LUT4_IN2 Register Field Descriptions.....	2718
29-27. CLB_LUT4_IN3 Register Field Descriptions.....	2719



29-28. CLB_FSM_LUT_FN1_0 Register Field Descriptions .....	2720
29-29. CLB_FSM_LUT_FN2 Register Field Descriptions .....	2721
29-30. CLB_LUT4_FN1_0 Register Field Descriptions.....	2722
29-31. CLB_LUT4_FN2 Register Field Descriptions.....	2723
29-32. CLB_FSM_NEXT_STATE_0 Register Field Descriptions .....	2724
29-33. CLB_FSM_NEXT_STATE_1 Register Field Descriptions .....	2725
29-34. CLB_FSM_NEXT_STATE_2 Register Field Descriptions .....	2726
29-35. CLB_MISC_CONTROL Register Field Descriptions .....	2727
29-36. CLB_OUTPUT_LUT_0 Register Field Descriptions .....	2730
29-37. CLB_OUTPUT_LUT_1 Register Field Descriptions .....	2731
29-38. CLB_OUTPUT_LUT_2 Register Field Descriptions .....	2732
29-39. CLB_OUTPUT_LUT_3 Register Field Descriptions .....	2733
29-40. CLB_OUTPUT_LUT_4 Register Field Descriptions .....	2734
29-41. CLB_OUTPUT_LUT_5 Register Field Descriptions .....	2735
29-42. CLB_OUTPUT_LUT_6 Register Field Descriptions .....	2736
29-43. CLB_OUTPUT_LUT_7 Register Field Descriptions .....	2737
29-44. CLB_HLC_EVENT_SEL Register Field Descriptions .....	2738
29-45. CLB_LOGIC_CONTROL_REGS Registers.....	2739
29-46. CLB_LOGIC_CONTROL_REGS Access Type Codes .....	2739
29-47. CLB_LOAD_EN Register Field Descriptions .....	2741
29-48. CLB_LOAD_ADDR Register Field Descriptions .....	2742
29-49. CLB_LOAD_DATA Register Field Descriptions .....	2743
29-50. CLB_INPUT_FILTER Register Field Descriptions .....	2744
29-51. CLB_IN_MUX_SEL_0 Register Field Descriptions .....	2746
29-52. CLB_LCL_MUX_SEL_1 Register Field Descriptions.....	2748
29-53. CLB_LCL_MUX_SEL_2 Register Field Descriptions.....	2749
29-54. CLB_BUF_PTR Register Field Descriptions.....	2750
29-55. CLB_GP_REG Register Field Descriptions.....	2751
29-56. CLB_OUT_EN Register Field Descriptions .....	2752
29-57. CLB_GLBL_MUX_SEL_1 Register Field Descriptions .....	2753
29-58. CLB_GLBL_MUX_SEL_2 Register Field Descriptions .....	2754
29-59. CLB_INTR_TAG_REG Register Field Descriptions .....	2755
29-60. CLB_LOCK Register Field Descriptions.....	2756
29-61. CLB_DBG_R0 Register Field Descriptions .....	2757
29-62. CLB_DBG_R1 Register Field Descriptions .....	2758
29-63. CLB_DBG_R2 Register Field Descriptions .....	2759
29-64. CLB_DBG_R3 Register Field Descriptions .....	2760
29-65. CLB_DBG_C0 Register Field Descriptions .....	2761
29-66. CLB_DBG_C1 Register Field Descriptions .....	2762
29-67. CLB_DBG_C2 Register Field Descriptions .....	2763
29-68. CLB_DBG_OUT Register Field Descriptions.....	2764
29-69. CLB_DATA_EXCHANGE_REGS Registers .....	2766
29-70. CLB_DATA_EXCHANGE_REGS Access Type Codes .....	2766
29-71. CLB_PUSH_y Register Field Descriptions.....	2767
29-72. CLB_PULL_y Register Field Descriptions .....	2768
29-73. CLB Registers to Driverlib Functions .....	2769

## About This Manual

---

---

This Technical Reference Manual (TRM) details the integration, the environment, the functional description, and the programming models for each peripheral and subsystem in the device.

The TRM should not be considered a substitute for the data manual, rather a companion guide that should be used alongside the device-specific data manual to understand the details to program the device. The primary purpose of the TRM is to abstract the programming details of the device from the data manual. This allows the data manual to outline the high-level features of the device without unnecessary information about register descriptions or programming models.

### Trademarks

C2000, Code Composer Studio are trademarks of Texas Instruments.

### Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers may be shown with the suffix h or the prefix 0x. For example, the following number is 40 hexadecimal (decimal 64): 40h or 0x40.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties with default reset value below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure can have one of multiple meanings:
    - Not implemented on the device
    - Reserved for future device expansion
    - Reserved for TI testing
    - Reserved configurations of the device that are not supported
  - Writing nondefault values to the Reserved bits could cause unexpected behavior and should be avoided.

### Glossary

*TI Glossary* — This glossary lists and explains terms, acronyms, and definitions.

### Related Documentation From Texas Instruments

For a complete listing of related documentation and development-support tools for these devices, visit the Texas Instruments website at <http://www.ti.com>. Additionally, the *TMS320C28x CPU and Instruction Set Reference Guide (SPRU430)* and *TMS320C28x Floating Point Unit and Instruction Set Reference Guide (SPRUE02)* must be used in conjunction with this TRM.

## C2000 Software Support

---

---

C2000Ware for C2000™ Microcontrollers is a cohesive set of development software and documentation designed to minimize software development time. From device-specific drivers and libraries to device peripheral examples, C2000Ware provides a solid foundation to begin development and evaluation of your product.

Topic	Page
1.1 Introduction .....	82
1.2 C2000Ware Structure .....	82
1.3 Documentation .....	82
1.4 Devices .....	82
1.5 Libraries.....	82
1.6 Code Composer Studio .....	82
1.7 PinMUX Tool.....	83

## 1.1 Introduction

C2000Ware for C2000™ Microcontrollers is a cohesive set of development software and documentation designed to minimize software development time. From device-specific drivers and libraries to device peripheral examples, C2000Ware provides a solid foundation to begin development and evaluation of your product.

C2000Ware can be downloaded from: [www.ti.com/tool/C2000WARE](http://www.ti.com/tool/C2000WARE)

## 1.2 C2000Ware Structure

The C2000Ware software package is organized into the following directory structure:

**Table 1-1. C2000Ware Root Directories**

Directory Name	Description
boards	Contains the hardware design schematics, BOM, gerber files, and documentation for C2000 controlCARDS,
device_support	Contains all device-specific support files, bit field headers and device development user's guides.
docs	Contains the C2000Ware package user's guides and the HTML index page of all package documentation.
driverlib	Contains the device-specific driver library and driver-based peripheral examples.
libraries	Contains the device-specific and core libraries.

## 1.3 Documentation

Within C2000Ware, there is an extensive amount of development documentation ranging from board design documentation, to library user's guides, to driver API documentation. The "boards" directory contains all the hardware design, BOM, gerber files, and more for controlCARDS. To assist with locating the necessary documentation, an HTML page is provided that contains a full list of all the documents in the C2000Ware package. Locate this page in the "docs" directory.

## 1.4 Devices

C2000Ware contains the necessary software and documentation to jumpstart development for C2000 microcontrollers. Each device includes device-specific common source files, peripheral example projects, bit field headers, and if available, a device peripheral driver library. Additionally, documentation is provided for each device on how to set up a CCS project, as well as give an overview of all the included example projects and assist with troubleshooting. For devices with a driver library, documentation is also included that details all the peripheral APIs available.

To learn more about C2000 microcontrollers, visit [www.ti.com/c2000](http://www.ti.com/c2000).

## 1.5 Libraries

The libraries included in C2000Ware range from fixed point and floating point math libraries, to specialized DSP libraries, as well as calibration libraries. Each library includes documentation and examples, where applicable. Additionally, the flash API files and boot ROM source code are located in the "libraries" directory.

## 1.6 Code Composer Studio

Code Composer Studio is an integrated development environment (IDE) that supports TI's microcontroller and embedded processors portfolio. Code Composer Studio comprises a suite of tools used to develop and debug embedded applications. The latest version of Code Composer Studio can be obtained at the following link:

<http://www.ti.com/ccstudio>

All projects and examples in C2000Ware are built for and tested with TI's Code Composer Studio. Although Code Composer Studio is not included with the C2000Ware installer, it is easily obtainable in a variety of versions.

## 1.7 PinMUX Tool

The Pin MUX Utility is a software tool which provides a Graphical User Interface for configuring pin multiplexing settings, resolving conflicts and specifying I/O cell characteristics for TI MPUs. Results are output as C header/code files that can be imported into software development kits (SDKs) or used to configure customer's custom software.

The latest version of the PinMux Tool can be obtained at the following link:<http://dev.ti.com/>

## C28x Processor

---



---

This chapter contains a short description of the C28x Processor and extended instruction sets. Further information about this device can be found in the following document(s):

[TMS320C28x CPU and Instruction Set Reference Guide](#)

[TMS320C28x Extended Instruction Sets Technical Reference Manual](#)

[Accelerators: Enhancing the Capabilities of the C2000 MCU Family Technical Brief](#)

[TMS320C28x FPU Primer](#)

Topic	Page
<b>2.1 Introduction</b> .....	<b>85</b>
<b>2.2 Features</b> .....	<b>85</b>
<b>2.3 Floating-Point Unit</b> .....	<b>85</b>
<b>2.4 Trigonometric Math Unit</b> .....	<b>85</b>
<b>2.5 Viterbi, Complex Math and CRC Unit (VCU)</b> .....	<b>86</b>

## 2.1 Introduction

The C28x CPU is a 32-bit fixed-point processor. This device draws from the best features of digital signal processing, reduced instruction set computing (RISC), microcontroller architectures, firmware, and tool sets.

For more information on CPU architecture and instruction set, see the [TMS320C28x CPU and Instruction Set Reference Guide](#).

## 2.2 Features

The CPU features include a modified Harvard architecture and circular addressing. The RISC features are single-cycle instruction execution, register-to-register operations, and modified Harvard architecture. The microcontroller features include ease of use through an intuitive instruction set, byte packing and unpacking, and bit manipulation. The modified Harvard architecture of the CPU enables instruction and data fetches to be performed in parallel. The CPU can read instructions and data while it writes data simultaneously to maintain the single-cycle instruction operation across the pipeline.

## 2.3 Floating-Point Unit

The C28x plus floating-point (C28x+FPU) processor extends the capabilities of the C28x fixed-point CPU by adding registers and instructions to support IEEE single-precision floating point operations.

Devices with the C28x+FPU include the standard C28x register set plus an additional set of floating-point unit registers. The additional floating-point unit registers are the following:

- Eight floating-point result registers, RnH (where n = 0–7)
- Floating-point Status Register (STF)
- Repeat Block Register (RB)

All of the floating-point registers, except the repeat block register, are shadowed. This shadowing can be used in high-priority interrupts for fast context save and restore of the floating-point registers.

For more information, see the [TMS320C28x Extended Instruction Sets Technical Reference Manual](#).

## 2.4 Trigonometric Math Unit

The TMU extends the capabilities of a C28x+FPU by adding instructions and leveraging existing FPU instructions to speed up the execution of common trigonometric and arithmetic operations listed in [Table 2-1](#).

**Table 2-1. TMU Supported Instructions**

INSTRUCTIONS	C EQUIVALENT OPERATION	PIPELINE CYCLES
MPY2PIF32 RaH,RbH	$a = b * 2\pi$	2/3
DIV2PIF32 RaH,RbH	$a = b / 2\pi$	2/3
DIVF32 RaH,RbH,RcH	$a = b/c$	5
SQRTF32 RaH,RbH	$a = \text{sqrt}(b)$	5
SINPUF32 RaH,RbH	$a = \sin(b*2\pi)$	4
COSPUF32 RaH,RbH	$a = \cos(b*2\pi)$	4
ATANPUF32 RaH,RbH	$a = \text{atan}(b)/2\pi$	4
QUADF32 RaH,RbH,RcH,RdH	Operation to assist in calculating ATANPU2	5

No changes have been made to existing instructions, pipeline or memory bus architecture. All TMU instructions use the existing FPU register set (R0H to R7H) to carry out their operations.

For more information, see the [TMS320C28x Extended Instruction Sets Technical Reference Manual](#).



## 2.5 Viterbi, Complex Math and CRC Unit (VCU)

The C28x with VCU (C28x+VCU) processor extends the capabilities of the C28x fixed-point or floating-point CPU by adding registers and instructions to support the following algorithm types:

- **Viterbi decoding**

Viterbi decoding is commonly used in baseband communications applications. The viterbi decode algorithm consists of three main parts: branch metric calculations, compare-select (viterbi butterfly), and a traceback operation. [Table 2-2](#) shows a summary of the VCU performance for each of these operations.

**Table 2-2. Viterbi Decode Performance**

Viterbi Operation	VCU Cycles
Branch Metric Calculation (code rate = 1/2)	1
Branch Metric Calculation (code rate = 1/3)	2p
Viterbi Butterfly (add-compare-select)	2 <sup>(1)</sup>
Traceback per Stage	3 <sup>(2)</sup>

<sup>(1)</sup> C28x CPU takes 15 cycles per butterfly.

<sup>(2)</sup> C28x CPU takes 22 cycles per stage.

- **Cyclic redundancy check (CRC)**

CRC algorithms provide a straightforward method for verifying data integrity over large data blocks, communication packets, or code sections. The C28x+VCU can perform 8-, 16-, and 32-bit CRCs. For example, the VCU can compute the CRC for a block length of 10 bytes in 10 cycles. A CRC result register contains the current CRC which is updated whenever a CRC instruction is executed.

- **Complex math**

Complex math is used in many applications; a few of which are:

- Fast fourier transform (FFT)

The complex FFT is used in spread spectrum communications, as well as many signal processing algorithms.

- Complex filters

Complex filters improve data reliability, transmission distance, and power efficiency. The C28x+VCU can perform a complex I and Q multiply with coefficients (four multiplies) in a single cycle. In addition, the C28x+VCU can read/write the real and imaginary parts of 16-bit complex data to memory in a single cycle.

[Table 2-3](#) shows a summary of a few complex math operations enabled by the VCU.

**Table 2-3. Complex Math Performance**

Complex Math Operation	VCU Cycles	Notes
Add Or Subtract	1	32 +/- 32 = 32-bit (Useful for filters)
Add or Subtract	1	16 +/- 32 = 15-bit (Useful for FFT)
Multiply	2p	16 x 16 = 32-bit
Multiply & Accumulate (MAC)	2p	32 + 32 = 32-bit, 16 x 16 = 32-bit
RPT MAC	2p+N	Repeat MAC. Single cycle after the first operation.

Throughout this document the following notations are used:

- C28x refers to the C28x fixed-point CPU.
- C28x plus Floating-Point and C28x+FPU both refer to the C28x CPU with enhancements to support IEEE single-precision floating-point operations.
- C28x plus VCU and C28x+VCU both refer to the C28x CPU with enhancements to support viterbi decode, complex math and CRC.
- Some devices have both the FPU and the VCU. These are referred to as C28x+FPU+VCU.

For more information, see the [TMS320C28x Extended Instruction Sets Technical Reference Manual](#).

## TMS320F28004x System Control

The system-level functionality of this microcontroller configures the clocking, resets, and interrupts of the CPU and peripherals, as well as the operation of the on-chip memories, timers, and security features.

Further information about this device can be found in the following document(s):

- [A Technical Introduction to the TMS320F28004x Microcontroller](#)
- [The TMS320F28004x MCU: A Comparison to the TMS320F2806x and TMS320F2803x MCUs](#)

Topic	Page
<b>3.1 Introduction</b> .....	<b>88</b>
<b>3.2 Power Management</b> .....	<b>88</b>
<b>3.3 Device Identification and Configuration Registers</b> .....	<b>89</b>
<b>3.4 Resets</b> .....	<b>89</b>
<b>3.5 Peripheral Interrupts</b> .....	<b>91</b>
<b>3.6 Exceptions and Non-Maskable Interrupts</b> .....	<b>102</b>
<b>3.7 Clocking</b> .....	<b>104</b>
<b>3.8 32-Bit CPU Timers 0/1/2</b> .....	<b>114</b>
<b>3.9 Watchdog Timer</b> .....	<b>115</b>
<b>3.10 Low Power Modes</b> .....	<b>118</b>
<b>3.11 Memory Controller Module</b> .....	<b>121</b>
<b>3.12 Flash and OTP Memory</b> .....	<b>128</b>
<b>3.13 Dual Code Security Module (DCSM)</b> .....	<b>139</b>
<b>3.14 F28004x System Control Registers</b> .....	<b>155</b>

## 3.1 Introduction

System-level configuration is controlled by a group of submodules which are collectively referred to as the system control module. The system control module provides the following capabilities:

- System-level resets, including power-on and brownout resets
- Clock source selection and PLL configuration
- Missing clock detection
- Clock-gating low-power modes
- Peripheral interrupt handling
- Non-maskable interrupts for certain fault conditions
- Three 32-bit timers
- Windowed watchdog timer, which can generate an interrupt or a reset
- RAM initialization, write protection, and mastership control
- Flash memory ECC, wait state, and cache configuration
- Dual-zone code security module

Some registers in the system are protected from spurious CPU writes by the EALLOW protection mechanism. This uses the special CPU instructions EALLOW and EDIS to enable and disable access to protected registers. The current protection state is given by the EALLOW bit in the CPU's ST1 register.

**Table 3-1. Access to EALLOW-Protected Registers**

EALLOW Bit	CPU Writes	CPU Reads	JTAG Writes	JTAG Reads
0	Ignored	Allowed	Allowed <sup>(1)</sup>	Allowed
1	Allowed	Allowed	Allowed	Allowed

<sup>(1)</sup> The EALLOW bit is overridden via the JTAG port, allowing full access of protected registers during debug from the Code Composer Studio interface.

Register protection is enabled by default at startup. While protected, all writes to protected registers by the CPU are ignored. Only CPU reads, JTAG reads, and JTAG writes are allowed. If protection is disabled by executing the EALLOW instruction, the CPU is allowed to write freely to protected registers. After modifying registers, they can once again be protected by executing the EDIS instruction to clear the EALLOW bit.

Writes to the clock configuration and peripheral clock enable registers may be disabled until the next reset by writing to special lock registers.

## 3.2 Power Management

### 3.2.1 Internal 1.2-V Switching Regulator (DC-DC)

The internal DC-DC regulator is disabled by default. To use this supply, the TMS320F28004x MCU core must power up initially with the internal LDO (VREG) and then transition to the internal DC-DC regulator via application software. Follow the procedure below to enable the internal DC-DC regulator:

1. Set EALLOW bit.
2. Set the DCDCCTL.DCDCEN bit.
3. Wait for DCDCSTS.SWSEQDON to equal 0x1.
  - If this bit does not get set then the transition from VREG to DC-DC was not successful (check the hardware connections on the application board). Consult the F28004x Data Manual for schematic details.
4. Wait for DCDCSTS.INDDTECT to equal 0x1.
  - If this bit does not get set then the inductor was not detected on the VSW pin.
  - Note that steps 3 and 4 can be combined.
5. Delay 80us for the DC-DC regulator output to settle (delay function based on 10MHz CPU clock).

## 6. EDIS

The procedure to enable the internal DC-DC regulator must be completed prior to all other initialization functions and application code.

### 3.3 Device Identification and Configuration Registers

The device identification registers and configuration registers provide information on the part number, product family, revision, pin count, qualification status, and feature availability of the device.

All of the device information is part of the DEV\_CFG\_REGS space. The identification registers are PARTIDL, PARTIDH, and REVID.

A 256-bit Unique ID (UID) is available in UID\_REGS. The 256 bits are separated into these registers:

- UID\_PSRAND0-5: 192 bits of pseudo-random data
- UID\_UNIQUE: 32-bit unique data, the value in this register will be unique across all devices with the same PARTIDH
- UID\_CHECKSUM: 32-bit fletcher checksum of UID\_PSRAND0-5 and UID\_UNIQUE and calculated as either little- or big-endian during factory testing

### 3.4 Resets

This section explains the types and effects of the different resets on this device.

#### 3.4.1 Reset Sources

[Table 3-2](#) summarizes the various reset signals and their effect on the device.

**Table 3-2. Reset Signals**

Reset Source	CPU Core Reset (C28x, FPU, VCU)	Peripherals Reset	JTAG / Debug Logic Reset	IOs	XRS Output
POR	Yes	Yes	Yes	Hi-Z	Yes
XRS Pin	Yes	Yes	No	Hi-Z	-
WDRS	Yes	Yes	No	Hi-Z	Yes
NMIWDRS	Yes	Yes	No	Hi-Z	Yes
SYSRS (Debugger Reset)	Yes	Yes	No	Hi-Z	No
SCCRESET	Yes	Yes	No	Hi-Z	No

The resets can be divided into two groups:

- Chip-level resets ( $\overline{XRS}$ , POR, BOR,  $\overline{WDRS}$ , and  $\overline{NMIWDRS}$ ), which reset all or almost all of the device.
- System resets ( $\overline{SYSRS}$  and  $\overline{SCCRESET}$ ), which reset a large subset of the device but maintain some system-level configuration.

After a reset, the reset cause register (RESC) is updated with the reset cause. The bits in this register maintain their state across multiple resets. They can only be cleared by a power-on reset (POR) or by writing ones to the RESCCLR register. Some are cleared by the boot ROM as part of its start-up routines.

Many peripheral modules have individual resets accessible through the SOFTPRESx registers. For information about a module's reset state, refer to the appropriate chapter for that module.

After any reset, the CPU begins execution from address 0x3FFFC0 (the reset vector), which is in the boot ROM. After running the boot ROM code, the CPU will typically branch to the start of the flash memory at address 0x80000. For more information on controlling the boot process, see the ROM Code and Peripheral Booting chapter.

---

**NOTE:** After a POR, the boot ROMs will clear the M0/M1, LSx, GSx, and message RAMs to ensure that they contain valid ECC or parity.

---

### 3.4.2 External Reset ( $\overline{XRS}$ )

The external reset ( $\overline{XRS}$ ) is the main chip-level reset for the device. It resets the CPU, all peripherals and I/O pin configurations, and most of the system control registers. There is a dedicated open-drain pin for  $\overline{XRS}$ . This pin may be used to drive reset pins for other ICs in the application, and may itself be driven by an external source. The  $\overline{XRS}$  is driven internally during watchdog, NMI, and power-on resets.

The XRSn bit in the RESC register will be set whenever  $\overline{XRS}$  is driven low for any reason. This bit is then cleared by the boot ROM.

### 3.4.3 Power-On Reset (POR)

The power-on reset (POR) circuit creates a clean reset throughout the device during power-up, suppressing glitches on the GPIOs. The  $\overline{XRS}$  pin is held low for the duration of the POR. In most applications,  $\overline{XRS}$  is held low long enough to reset other system ICs, but some applications may require a longer pulse. In these cases, the  $\overline{XRS}$  pin can be driven low externally to provide the correct reset duration. A POR resets everything that  $\overline{XRS}$  does, along with a few other registers – the reset cause register (RESC), the NMI shadow flag register (NMISHDFLG), and the X1 clock counter register (X1CNT). A POR also resets the debug logic used by the JTAG port.

After a POR, the POR and XRSn bits in RESC are set. These bits are then cleared by the boot ROM.

### 3.4.4 Debugger Reset ( $\overline{SYSRS}$ )

During development, it is sometimes necessary to reset the CPU and its peripherals without disconnecting the debugger or disrupting the system-level configuration. To facilitate this, the CPU has its own subsystem reset, which can be triggered by a debugger using Code Composer Studio. This reset ( $\overline{SYSRS}$ ) resets the CPU, its peripherals, many system control registers (including its clock gating and LPM configuration), and all I/O pin configurations.

The  $\overline{SYSRS}$  does not reset the ICEPick debug module, the device capability registers, the clock source and PLL configurations, the missing clock detection state, the PIE vector fetch error handler address, the NMI flags, the analog trims, or anything reset only by a POR (see [Section 3.4.3](#)).

### 3.4.5 Watchdog Reset ( $\overline{WDRS}$ )

The device has a watchdog timer that can optionally trigger a reset if it is not serviced by the CPU within a user-specified amount of time. This watchdog reset ( $\overline{WDRS}$ ) produces an  $\overline{XRS}$  that lasts for 512 INTOSC1 cycles.

After a watchdog reset, the WDRSn bit in RESC is set.

### 3.4.6 NMI Watchdog Reset ( $\overline{NMIWDRS}$ )

The device has a non-maskable interrupt (NMI) module that detects hardware errors in the system. The NMI module has a watchdog timer that triggers a reset if the CPU does not respond to an error within a user-specified amount of time. This NMI watchdog reset ( $\overline{NMIWDRS}$ ) produces an  $\overline{XRS}$  that lasts for 512 INTOSC1 cycles.

After an NMI watchdog reset, the NMIWDRSn bit in RESC is set.

### 3.4.7 DCSM Safe Code Copy Reset ( $\overline{SCCRESET}$ )

The device has a dual-zone code security module (DCSM) that blocks read access to certain areas of the flash memory. To facilitate CRC checks and copying of CLA code, TI provides ROM functions to securely access those memory areas. To prevent security breaches, interrupts must be disabled before calling these functions. If a vector fetch occurs in a safe copy or CRC function, the DCSM triggers a reset. This security reset ( $\overline{SCCRESET}$ ) is similar to a  $\overline{SYSRS}$ . However, the security reset also resets the debug logic to deny access to a potential attacker.

After a security reset, the SCCRESETn bit in RESC is set.

### 3.5 Peripheral Interrupts

This section explains the peripheral interrupt handling on the device. Non-maskable interrupts are covered in [Section 3.6](#). Software interrupts and emulation interrupts are not covered in this document. For information on those, see the [TMS320C28x CPU and Instruction Set Reference Guide](#).

#### 3.5.1 Interrupt Concepts

An interrupt is a signal that causes the CPU to pause its current execution and branch to a different piece of code known as an interrupt service routine (ISR). This is a useful mechanism for handling peripheral events, and involves less CPU overhead or program complexity than register polling. However, because interrupts are asynchronous to the program flow, care must be taken to avoid conflicts over resources that are accessed both in interrupts and in the main program code.

Interrupts propagate to the CPU through a series of flag and enable registers. The flag registers store the interrupt until it is processed. The enable registers block the propagation of the interrupt. When an interrupt signal reaches the CPU, the CPU fetches the appropriate ISR address from a list called the vector table.

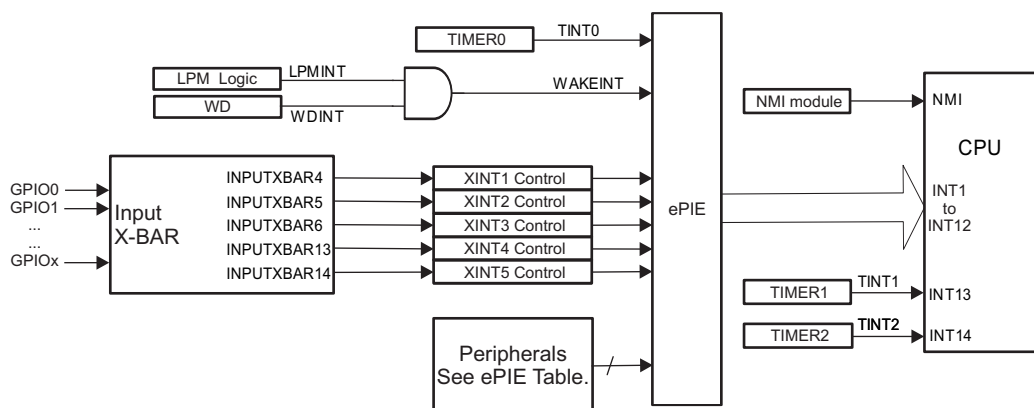
#### 3.5.2 Interrupt Architecture

The C28x CPU has fourteen peripheral interrupt lines. Two of them (INT13 and INT14) are connected directly to CPU timers 1 and 2, respectively. The remaining twelve are connected to peripheral interrupt signals through the enhanced Peripheral Interrupt Expansion module (ePIE, or PIE as a shortened version). The PIE multiplexes up to sixteen peripheral interrupts into each CPU interrupt line. It also expands the vector table to allow each interrupt to have its own ISR. This allows the CPU to support a large number of peripherals.

An interrupt path is divided into three stages – the peripheral, the PIE, and the CPU. Each stage has its own enable and flag registers. This system allows the CPU to handle one interrupt while others are pending, implement and prioritize nested interrupts in software, and disable interrupts during certain critical tasks.

Figure 3-1 shows the interrupt architecture for this device.

Figure 3-1. Device Interrupt Architecture



##### 3.5.2.1 Peripheral Stage

Each peripheral has its own unique interrupt configuration, which is described in that peripheral's chapter. Some peripherals allow multiple events to trigger the same interrupt signal. For example, a communications peripheral might use the same interrupt to indicate that data has been received or that there has been a transmission error. The cause of the interrupt can be determined by reading the peripheral's status register. Often, the bits in the status register must be cleared manually before another interrupt will be generated.

### 3.5.2.2 PIE Stage

The PIE provides individual flag and enable register bits for each of the peripheral interrupt signals, which are sometimes called PIE channels. These channels are grouped according to their associated CPU interrupt. Each PIE group has one 16-bit enable register (PIEIERx), one 16-bit flag register (PIEIFRx), and one bit in the PIE acknowledge register (PIEACK). The PIEACK register bit acts as a common interrupt mask for the entire PIE group.

When the CPU receives an interrupt, it fetches the address of the ISR from the PIE. The PIE returns the vector for the lowest-numbered channel in the group that is both flagged and enabled. This gives lower-numbered interrupts a higher priority when multiple interrupts are pending.

If no interrupt is both flagged and enabled, the PIE returns the vector for channel 1. This condition will not happen unless software changes the state of the PIE while an interrupt is propagating. [Section 3.5.4](#) contains procedures for safely modifying the PIE configuration once interrupts have been enabled.

### 3.5.2.3 CPU Stage

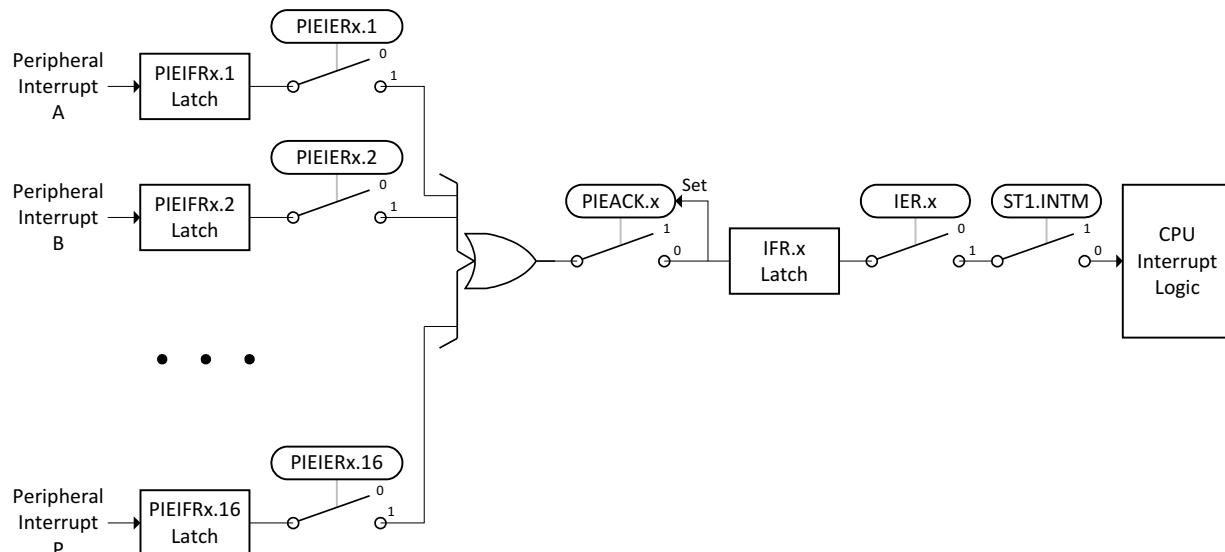
Like the PIE, the CPU provides flag and enable register bits for each of its interrupts. There is one enable register (IER) and one flag register (IFR), both of which are internal CPU registers. There is also a global interrupt mask, which is controlled by the INTM bit in the ST1 register. This mask can be set and cleared using the CPU's SETC and CLRC instructions. In C code, C2000Ware's DINT and EINT macros can be used for this purpose.

Writes to IER and INTM are atomic operations. In particular, if INTM is set, the next instruction in the pipeline will run with interrupts disabled. No software delays are needed.

## 3.5.3 Interrupt Entry Sequence

[Figure 3-2](#) shows how peripheral interrupts propagate to the CPU.

**Figure 3-2. Interrupt Propagation Path**



When a peripheral generates an interrupt (on PIE group x, channel y), it triggers the following sequence of events:

1. The interrupt is latched in PIEIFRx.y.
2. If PIEIERx.y is set, the interrupt propagates.
3. If PIEACK.x is clear, the interrupt propagates and PIEACK.x is set.
4. The interrupt is latched in IFR.x.
5. If IER.x is set, the interrupt propagates.
6. If INTM is clear, the CPU receives the interrupt.



7. Any instructions in the D2 or later stage of the pipeline are run to completion. Instructions in earlier stages are flushed.
8. The CPU saves its context on the stack.
9. IFR.x and IER.x are cleared. INTM is set. EALLOW is cleared.
10. The CPU fetches the ISR vector from the PIE. PIEIFRx.y is cleared.
11. The CPU branches to the ISR.

The interrupt latency is the time between PIEIFRx.y latching the interrupt and the first ISR instruction entering the execution stage of the CPU pipeline. The minimum interrupt latency is 14 SYSCLK cycles. Wait states on the ISR or stack memories will add to the latency. External interrupts add a minimum of two SYSCLK cycles for GPIO synchronization plus extra time for input qualification (if used). Loops created using the C28x RPT instruction cannot be interrupted.

### 3.5.4 Configuring and Using Interrupts

At power-up, no interrupts are enabled by default. The PIEIER and IER registers are cleared and INTM is set. The application code is responsible for configuring and enabling all peripheral interrupts.

#### 3.5.4.1 Enabling Interrupts

To enable a peripheral interrupt, perform the following steps:

1. Disable interrupts globally (DINT or SETC INTM).
2. Enable the PIE by setting the ENPIE bit of the PIECTRL register.
3. Write the ISR vector for each interrupt to the appropriate location in the PIE vector table, which can be found in [Table 3-3](#). Note that the vector table is EALLOW-protected.
4. Set the appropriate PIEIERx bit for each interrupt. The PIE group and channel assignments can be found in [Table 3-3](#).
5. Set the CPU IER bit for any PIE group containing enabled interrupts.
6. Enable the interrupt in the peripheral.
7. Enable interrupts globally (EINT or CLRC INTM).

Step 4 does not apply to the Timer1 and Timer2 interrupts, which connect directly to the CPU.

#### 3.5.4.2 Handling Interrupts

ISRs are similar to normal functions, but must do the following:

1. Save and restore the state of certain CPU registers (if used).
2. Clear the PIEACK bit for the interrupt group.
3. Return using the IRET instruction.

Requirements 1 and 3 are handled automatically by the TMS320C28x C compiler if the function is defined using the `__interrupt` keyword. For information on this keyword, see the Keywords section of the [TMS320C28x Optimizing C/C++ Compiler v6.2.4 User's Guide](#). For information on writing assembly code to handle interrupts, see the Standard Operation for Maskable Interrupts section of the [TMS320C28x CPU and Instruction Set Reference Guide](#).

The PIEACK bit for the interrupt group must be cleared manually in user code. This is normally done at the end of the ISR. If the PIEACK bit is not cleared, the CPU will not receive any further interrupts from that group. This does not apply to the Timer1 and Timer2 interrupts, which do not go through the PIE.

#### 3.5.4.3 Disabling Interrupts

To disable all interrupts, set the CPU's global interrupt mask via DINT or SETC INTM. It is not necessary to add NOPs after setting INTM or modifying IER – the next instruction will execute with interrupts disabled.

Individual interrupts can be disabled using the PIEIERx registers, but care must be taken to avoid race conditions. If an interrupt signal is already propagating when the PIEIER write completes, it may reach the CPU and trigger a spurious interrupt condition. To avoid this, use the following procedure:

1. Disable interrupts globally (DINT or SETC INTM).
2. Clear the PIEIER bit for the interrupt.
3. Wait 5 cycles to make sure that any propagating interrupt has reached the CPU IFR register.
4. Clear the CPU IFR bit for the interrupt's PIE group.
5. Clear the PIEACK bit for the interrupt's PIE group.
6. Enable interrupts globally (EINT or CLRC INTM).

Interrupt groups can be disabled using the CPU IER register. This cannot cause a race condition, so no special procedure is needed.

PIEIFR bits must never be cleared in software since the read/modify/write operation may cause incoming interrupts to be lost. The only safe way to clear a PIEIFR bit is to have the CPU take the interrupt. The following procedure can be used to bypass the normal ISR:

1. Disable interrupts globally (DINT or SETC INTM).
2. Modify the PIE vector table to map the PIEIFR bit's interrupt vector to an empty ISR. This ISR will only contain a return from interrupt instruction (IRET).
3. Disable the interrupt in the peripheral registers.
4. Enable interrupts globally (EINT or CLRC INTM).
5. Wait for the pending interrupt to be serviced by the empty ISR.
6. Disable interrupts globally.
7. Modify the PIE vector table to map the interrupt vector back to its original ISR.
8. Clear the PIEACK bit for the interrupt's PIE group.
9. Enable interrupts globally.

#### 3.5.4.4 Nesting Interrupts

By default, interrupts do not nest. It is possible to nest and prioritize interrupts via software control of the IER and PIEIERx registers. Documentation and example code can be found in C2000Ware and on the TI Processors wiki: [http://processors.wiki.ti.com/index.php/Interrupt\\_Nesting\\_on\\_C28x](http://processors.wiki.ti.com/index.php/Interrupt_Nesting_on_C28x)

#### 3.5.4.5 Vector Address Validity Check

There are two copies of the ePIE vector table. The primary vector table is located at addresses 0xD00 - 0xEFF. The redundant vector table is located at addresses 0x01000D00 - 0x01000EFF. A write to a primary vector address writes to both tables, while a write to a redundant vector address only writes to the redundant table. Both tables are read independently.

During a vector fetch, the ePIE performs a hardware comparison of both vector table outputs. If there is a mismatch between the two vector tables, the CPU branches to the address in the PIEVERRADDR register and the ePIE sends trip signals to the PWMs. If the PIEVERRADDR register value has not been set, the default boot ROM handler at address 0x003FFFBE is used.

### 3.5.5 PIE Channel Mapping

Table 3-3 shows the PIE group and channel assignments for each peripheral interrupt. Each row is a group, and each column is a channel within that group. When multiple interrupts are pending, the lowest-numbered channel is the lowest-numbered group is serviced first. Thus, the interrupts at the top of the table have the highest priority, and the interrupts at the bottom have the lowest priority.

**Table 3-3. PIE Channel Mapping**

	INTx.1	INTx.2	INTx.3	INTx.4	INTx.5	INTx.6	INTx.7	INTx.8	INTx.9	INTx.10	INTx.11	INTx.12	INTx.13	INTx.14	INTx.15	INTx.16
INT1.y	ADCA1	ADCB1	ADCC1	XINT1	XINT2	-	TIMER0	WAKE/WDOG	-	-	-	-	-	-	-	-
INT2.y	EPWM1_TZ	EPWM2_TZ	EPWM3_TZ	EPWM4_TZ	EPWM5_TZ	EPWM6_TZ	EPWM7_TZ	EPWM8_TZ	-	-	-	-	-	-	-	-
INT3.y	EPWM1	EPWM2	EPWM3	EPWM4	EPWM5	EPWM6	EPWM7	EPWM8	-	-	-	-	-	-	-	-
INT4.y	ECAP1	ECAP2	ECAP3	ECAP4	ECAP5	ECAP6	ECAP7	-	-	-	-	-	-	ECAP6_HRCAL	ECAP7_HRCAL	-
INT5.y	EQEP1	EQEP2	-	-	CLB1	CLB2	CLB3	CLB4	SDFM1	-	-	-	SDFM1DR1	SDFM1DR2	SDFM1DR3	SDFM1DR4
INT6.y	SPIA_RX	SPIA_TX	SPIB_RX	SPIB_TX	-	-	-	-	-	-	-	-	-	-	-	-
INT7.y	DMA_CH1	DMA_CH2	DMA_CH3	DMA_CH4	DMA_CH5	DMA_CH6	-	-	-	-	FSITX_INT1	FSITX_INT2	FSIRX_INT1	FSIRX_INT2	CLAPROMCRC	DCC
INT8.y	I2CA	I2CA_FIFO	-	-	-	-	-	-	LINA_0	LINA_1	-	-	PMBUSA	-	-	-
INT9.y	SCIA_RX	SCIA_TX	SCIB_RX	SCIB_TX	CANA_0	CANA_1	CANB_0	CANB_1	-	-	-	-	-	-	-	-
INT10.y	ADCA_EVT	ADCA2	ADCA3	ADCA4	ADCB_EVT	ADCB2	ADCB3	ADCB4	ADCC_EVT	ADCC2	ADCC3	ADCC4	-	-	-	-
INT11.y	CLA1_1	CLA1_2	CLA1_3	CLA1_4	CLA1_5	CLA1_6	CLA1_7	CLA1_8	-	-	-	-	-	-	-	-
INT12.y	XINT3	XINT4	XINT5	-	-	-	FPU_OVERFLOW	FPU_UNDERFLOW	-	RAM_CORRECTABLE_ERROR	FLASH_CORRECTABLE_ERROR	RAM_ACCESS_VIOLATION	SYS_PLL_SLIP	-	CLA_OVERFLOW	CLA_UNDERFLOW

Note: Cells marked "-" are Reserved

### 3.5.6 Vector Tables

Table 3-4 shows the CPU interrupt vector table. The vectors for INT1 – INT12 are not used in this device. The reset vector is fetched from the boot ROM instead of from this table. All vectors are EALLOW-protected.

**Table 3-4. CPU Interrupt Vectors**

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
Reset	0	0x0000 0D00	2	Reset is always fetched from location 0x003F_FFC0 in Boot ROM	1 (Highest)	-
INT1	1	0x0000 0D02	2	Not used. See PIE Group 1	5	-
INT2	2	0x0000 0D04	2	Not used. See PIE Group 2	6	-
INT3	3	0x0000 0D06	2	Not used. See PIE Group 3	7	-
INT4	4	0x0000 0D08	2	Not used. See PIE Group 4	8	-
INT5	5	0x0000 0D0A	2	Not used. See PIE Group 5	9	-
INT6	6	0x0000 0D0C	2	Not used. See PIE Group 6	10	-
INT7	7	0x0000 0D0E	2	Not used. See PIE Group 7	11	-
INT8	8	0x0000 0D10	2	Not used. See PIE Group 8	12	-
INT9	9	0x0000 0D12	2	Not used. See PIE Group 9	13	-
INT10	10	0x0000 0D14	2	Not used. See PIE Group 10	14	-
INT11	11	0x0000 0D16	2	Not used. See PIE Group 11	15	-
INT12	12	0x0000 0D18	2	Not used. See PIE Group 12	16	-
INT13	13	0x0000 0D1A	2	CPU TIMER1 Interrupt	17	-
INT14	14	0x0000 0D1C	2	CPU TIMER2 Interrupt (for TI/RTOS use)	18	-
DATALOG	15	0x0000 0D1E	2	CPU Data Logging Interrupt	19 (lowest)	-
RTOSINT	16	0x0000 0D20	2	CPU Real-Time OS Interrupt	4	-
EMUINT	17	0x0000 0D22	2	CPU Emulation Interrupt	2	-
NMI	18	0x0000 0D24	2	Non-Maskable Interrupt	3	-
ILLEGAL	19	0x0000 0D26	2	Illegal Instruction (ITRAP)	-	-
USER 1	20	0x0000 0D28	2	User-Defined Trap	-	-
USER 2	21	0x0000 0D2A	2	User-Defined Trap	-	-
USER 3	22	0x0000 0D2C	2	User-Defined Trap	-	-
USER 4	23	0x0000 0D2E	2	User-Defined Trap	-	-
USER 5	24	0x0000 0D30	2	User-Defined Trap	-	-
USER 6	25	0x0000 0D32	2	User-Defined Trap	-	-
USER 7	26	0x0000 0D34	2	User-Defined Trap	-	-
USER 8	27	0x0000 0D36	2	User-Defined Trap	-	-
USER 9	28	0x0000 0D38	2	User-Defined Trap	-	-
USER 10	29	0x0000 0D3A	2	User-Defined Trap	-	-
USER 11	30	0x0000 0D3C	2	User-Defined Trap	-	-
USER 12	31	0x0000 0D3E	2	User-Defined Trap	-	-

Table 3-5 shows the pie vector table.

**Table 3-5. PIE Interrupt Vectors**

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
<b>PIE Group 1 Vectors - Muxed into CPU INT1</b>						
INT1.1	32	0x0000 0D40	2	ADCA1 interrupt	5	1 (Highest)
INT1.2	33	0x0000 0D42	2	ADCB1 interrupt	5	2
INT1.3	34	0x0000 0D44	2	ADCC1 interrupt	5	3
INT1.4	35	0x0000 0D46	2	XINT1 interrupt	5	4
INT1.5	36	0x0000 0D48	2	XINT2 interrupt	5	5
INT1.6	37	0x0000 0D4A	2	Reserved	5	6
INT1.7	38	0x0000 0D4C	2	TIMER0 interrupt	5	7
INT1.8	39	0x0000 0D4E	2	WAKE interrupt	5	8
INT1.9	128	0x0000 0E00	2	Reserved	5	9
INT1.10	129	0x0000 0E02	2	Reserved	5	10
INT1.11	130	0x0000 0E04	2	Reserved	5	11
INT1.12	131	0x0000 0E06	2	Reserved	5	12
INT1.13	132	0x0000 0E08	2	Reserved	5	13
INT1.14	133	0x0000 0E0A	2	Reserved	5	14
INT1.15	134	0x0000 0E0C	2	Reserved	5	15
INT1.16	135	0x0000 0E0E	2	Reserved	5	16 (Lowest)
<b>PIE Group 2 Vectors - Muxed into CPU INT2</b>						
INT2.1	40	0x0000 0D50	2	EPWM1 trip zone interrupt	6	1 (Highest)
INT2.2	41	0x0000 0D52	2	EPWM2 trip zone interrupt	6	2
INT2.3	42	0x0000 0D54	2	EPWM3 trip zone interrupt	6	3
INT2.4	43	0x0000 0D56	2	EPWM4 trip zone interrupt	6	4
INT2.5	44	0x0000 0D58	2	EPWM5 trip zone interrupt	6	5
INT2.6	45	0x0000 0D5A	2	EPWM6 trip zone interrupt	6	6
INT2.7	46	0x0000 0D5C	2	EPWM7 trip zone interrupt	6	7
INT2.8	47	0x0000 0D5E	2	EPWM8 trip zone interrupt	6	8
INT2.9	136	0x0000 0E10	2	Reserved	6	9
INT2.10	137	0x0000 0E12	2	Reserved	6	10
INT2.11	138	0x0000 0E14	2	Reserved	6	11
INT2.12	139	0x0000 0E16	2	Reserved	6	12
INT2.13	140	0x0000 0E18	2	Reserved	6	13
INT2.14	141	0x0000 0E1A	2	Reserved	6	14
INT2.15	142	0x0000 0E1C	2	Reserved	6	15
INT2.16	143	0x0000 0E1E	2	Reserved	6	16 (Lowest)
<b>PIE Group 3 Vectors - Muxed into CPU INT3</b>						
INT3.1	48	0x0000 0D60	2	EPWM1 interrupt	7	1 (Highest)
INT3.2	49	0x0000 0D62	2	EPWM2 interrupt	7	2
INT3.3	50	0x0000 0D64	2	EPWM3 interrupt	7	3
INT3.4	51	0x0000 0D66	2	EPWM4 interrupt	7	4
INT3.5	52	0x0000 0D68	2	EPWM5 interrupt	7	5

**Table 3-5. PIE Interrupt Vectors (continued)**

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
INT3.6	53	0x0000 0D6A	2	EPWM6 interrupt	7	6
INT3.7	54	0x0000 0D6C	2	EPWM7 interrupt	7	7
INT3.8	55	0x0000 0D6E	2	EPWM8 interrupt	7	8
INT3.9	144	0x0000 0E20	2	Reserved	7	9
INT3.10	145	0x0000 0E22	2	Reserved	7	10
INT3.11	146	0x0000 0E24	2	Reserved	7	11
INT3.12	147	0x0000 0E26	2	Reserved	7	12
INT3.13	148	0x0000 0E28	2	Reserved	7	13
INT3.14	149	0x0000 0E2A	2	Reserved	7	14
INT3.15	150	0x0000 0E2C	2	Reserved	7	15
INT3.16	151	0x0000 0E2E	2	Reserved	7	16 (Lowest)
<b>PIE Group 4 Vectors - Muxed into CPU INT4</b>						
INT4.1	56	0x0000 0D70	2	ECAP1 interrupt	8	1 (Highest)
INT4.2	57	0x0000 0D72	2	ECAP2 interrupt	8	2
INT4.3	58	0x0000 0D74	2	ECAP3 interrupt	8	3
INT4.4	59	0x0000 0D76	2	ECAP4 interrupt	8	4
INT4.5	60	0x0000 0D78	2	ECAP5 interrupt	8	5
INT4.6	61	0x0000 0D7A	2	ECAP6 interrupt	8	6
INT4.7	62	0x0000 0D7C	2	ECAP7 interrupt	8	7
INT4.8	63	0x0000 0D7E	2	Reserved	8	8
INT4.9	152	0x0000 0E30	2	Reserved	8	9
INT4.10	153	0x0000 0E32	2	Reserved	8	10
INT4.11	154	0x0000 0E34	2	Reserved	8	11
INT4.12	155	0x0000 0E36	2	Reserved	8	12
INT4.13	156	0x0000 0E38	2	Reserved	8	13
INT4.14	157	0x0000 0E3A	2	ECAP6 HR calibration interrupt	8	14
INT4.15	158	0x0000 0E3C	2	ECAP7 HR calibration interrupt	8	15
INT4.16	159	0x0000 0E3E	2	Reserved	8	16 (Lowest)
<b>PIE Group 5 Vectors - Muxed into CPU INT5</b>						
INT5.1	64	0x0000 0D80	2	EQEP1 interrupt	9	1 (Highest)
INT5.2	65	0x0000 0D82	2	EQEP2 interrupt	9	2
INT5.3	66	0x0000 0D84	2	Reserved	9	3
INT5.4	67	0x0000 0D86	2	Reserved	9	4
INT5.5	68	0x0000 0D88	2	CLB1 interrupt	9	5
INT5.6	69	0x0000 0D8A	2	CLB2 interrupt	9	6
INT5.7	70	0x0000 0D8C	2	CLB3 interrupt	9	7
INT5.8	71	0x0000 0D8E	2	CLB4 interrupt	9	8
INT5.9	160	0x0000 0E40	2	SDFM1 interrupt	9	9
INT5.10	161	0x0000 0E42	2	Reserved	9	10
INT5.11	162	0x0000 0E44	2	Reserved	9	11
INT5.12	163	0x0000 0E46	2	Reserved	9	12
INT5.13	164	0x0000 0E48	2	SDFM1 DR interrupt 1	9	13

**Table 3-5. PIE Interrupt Vectors (continued)**

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
INT5.14	165	0x0000 0E4A	2	SDFM1 DR interrupt 2	9	14
INT5.15	166	0x0000 0E4C	2	SDFM1 DR interrupt 3	9	15
INT5.16	167	0x0000 0E4E	2	SDFM1 DR interrupt 4	9	16 (Lowest)
<b>PIE Group 6 Vectors - Muxed into CPU INT6</b>						
INT6.1	72	0x0000 0D90	2	SPIA_RX interrupt	10	1 (Highest)
INT6.2	73	0x0000 0D92	2	SPIA_TX interrupt	10	2
INT6.3	74	0x0000 0D94	2	SPIB_RX interrupt	10	3
INT6.4	75	0x0000 0D96	2	SPIB_TX interrupt	10	4
INT6.5	76	0x0000 0D98	2	Reserved	10	5
INT6.6	77	0x0000 0D9A	2	Reserved	10	6
INT6.7	78	0x0000 0D9C	2	Reserved	10	7
INT6.8	79	0x0000 0D9E	2	Reserved	10	8
INT6.9	168	0x0000 0E50	2	Reserved	10	9
INT6.10	169	0x0000 0E52	2	Reserved	10	10
INT6.11	170	0x0000 0E54	2	Reserved	10	11
INT6.12	171	0x0000 0E56	2	Reserved	10	12
INT6.13	172	0x0000 0E58	2	Reserved	10	13
INT6.14	173	0x0000 0E5A	2	Reserved	10	14
INT6.15	174	0x0000 0E5C	2	Reserved	10	15
INT6.16	175	0x0000 0E5E	2	Reserved	10	16 (Lowest)
<b>PIE Group 7 Vectors - Muxed into CPU INT7</b>						
INT7.1	80	0x0000 0DA0	2	DMA_CH1 interrupt	11	1 (Highest)
INT7.2	81	0x0000 0DA2	2	DMA_CH2 interrupt	11	2
INT7.3	82	0x0000 0DA4	2	DMA_CH3 interrupt	11	3
INT7.4	83	0x0000 0DA6	2	DMA_CH4 interrupt	11	4
INT7.5	84	0x0000 0DA8	2	DMA_CH5 interrupt	11	5
INT7.6	85	0x0000 0DAA	2	DMA_CH6 interrupt	11	6
INT7.7	86	0x0000 0DAC	2	Reserved	11	7
INT7.8	87	0x0000 0DAE	2	Reserved	11	8
INT7.9	176	0x0000 0E60	2	Reserved	11	9
INT7.10	177	0x0000 0E62	2	Reserved	11	10
INT7.11	178	0x0000 0E64	2	FSITX_INT1	11	11
INT7.12	179	0x0000 0E66	2	FSITX_INT2	11	12
INT7.13	180	0x0000 0E68	2	FSIRX_INT1	11	13
INT7.14	181	0x0000 0E6A	2	FSIRX_INT2	11	14
INT7.15	182	0x0000 0E6C	2	CLAPROMCRC interrupt	11	15
INT7.16	183	0x0000 0E6E	2	Reserved	11	16 (Lowest)



**Table 3-5. PIE Interrupt Vectors (continued)**

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
<b>PIE Group 8 Vectors - Muxed into CPU INT8</b>						
INT8.1	88	0x0000 0DB0	2	I2CA interrupt	12	1 (Highest)
INT8.2	89	0x0000 0DB2	2	I2CA FIFO interrupt	12	2
INT8.3	90	0x0000 0DB4	2	Reserved	12	3
INT8.4	91	0x0000 0DB6	2	Reserved	12	4
INT8.5	92	0x0000 0DB8	2	Reserved	12	5
INT8.6	93	0x0000 0DBA	2	Reserved	12	6
INT8.7	94	0x0000 0DBC	2	Reserved	12	7
INT8.8	95	0x0000 0DBE	2	Reserved	12	8
INT8.9	184	0x0000 0E70	2	LINA interrupt 0	12	9
INT8.10	185	0x0000 0E72	2	LINA interrupt 1	12	10
INT8.11	186	0x0000 0E74	2	Reserved	12	11
INT8.12	187	0x0000 0E76	2	Reserved	12	12
INT8.13	188	0x0000 0E78	2	PMBUSA interrupt	12	13
INT8.14	189	0x0000 0E7A	2	Reserved	12	14
INT8.15	190	0x0000 0E7C	2	Reserved	12	15
INT8.16	191	0x0000 0E7E	2	Reserved	12	16 (Lowest)
<b>PIE Group 9 Vectors - Muxed into CPU INT9</b>						
INT9.1	96	0x0000 0DC0	2	SCIA RX interrupt	13	1 (Highest)
INT9.2	97	0x0000 0DC2	2	SCIA TX interrupt	13	2
INT9.3	98	0x0000 0DC4	2	SCIB RX interrupt	13	3
INT9.4	99	0x0000 0DC6	2	SCIB TX interrupt	13	4
INT9.5	100	0x0000 0DC8	2	CANA interrupt 0	13	5
INT9.6	101	0x0000 0DCA	2	CANA interrupt 1	13	6
INT9.7	102	0x0000 0DCC	2	CANB interrupt 0	13	7
INT9.8	103	0x0000 0DCE	2	CANB interrupt 1	13	8
INT9.9	192	0x0000 0E80	2	Reserved	13	9
INT9.10	193	0x0000 0E82	2	Reserved	13	10
INT9.11	194	0x0000 0E84	2	Reserved	13	11
INT9.12	195	0x0000 0E86	2	Reserved	13	12
INT9.13	196	0x0000 0E88	2	Reserved	13	13
INT9.14	197	0x0000 0E8A	2	Reserved	13	14
INT9.15	198	0x0000 0E8C	2	Reserved	13	15
INT9.16	199	0x0000 0E8E	2	Reserved	13	16 (Lowest)
<b>PIE Group 10 Vectors - Muxed into CPU INT10</b>						
INT10.1	104	0x0000 0DD0	2	ADCA event interrupt	14	1 (Highest)
INT10.2	105	0x0000 0DD2	2	ADCA2 interrupt	14	2
INT10.3	106	0x0000 0DD4	2	ADCA3 interrupt	14	3
INT10.4	107	0x0000 0DD6	2	ADCA4 interrupt	14	4
INT10.5	108	0x0000 0DD8	2	ADCB event interrupt	14	5
INT10.6	109	0x0000 0DDA	2	ADCB2 interrupt	14	6

**Table 3-5. PIE Interrupt Vectors (continued)**

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
INT10.7	110	0x0000 0DDC	2	ADCB3 interrupt	14	7
INT10.8	111	0x0000 0DDE	2	ADCB4 interrupt	14	8
INT10.9	200	0x0000 0E90	2	ADCC event interrupt	14	9
INT10.10	201	0x0000 0E92	2	ADCC2 interrupt	14	10
INT10.11	202	0x0000 0E94	2	ADCC3 interrupt	14	11
INT10.12	203	0x0000 0E96	2	ADCC4 interrupt	14	12
INT10.13	204	0x0000 0E98	2	Reserved	14	13
INT10.14	205	0x0000 0E9A	2	Reserved	14	14
INT10.15	206	0x0000 0E9C	2	Reserved	14	15
INT10.16	207	0x0000 0E9E	2	Reserved	14	16 (Lowest)
<b>PIE Group 11 Vectors - Muxed into CPU INT11</b>						
INT11.1	112	0x0000 0DE0	2	CLA interrupt 1	15	1 (Highest)
INT11.2	113	0x0000 0DE2	2	CLA interrupt 2	15	2
INT11.3	114	0x0000 0DE4	2	CLA interrupt 3	15	3
INT11.4	115	0x0000 0DE6	2	CLA interrupt 4	15	4
INT11.5	116	0x0000 0DE8	2	CLA interrupt 5	15	5
INT11.6	117	0x0000 0DEA	2	CLA interrupt 6	15	6
INT11.7	118	0x0000 0DEC	2	CLA interrupt 7	15	7
INT11.8	119	0x0000 0DEE	2	CLA interrupt 8	15	8
INT11.9	208	0x0000 0EA0	2	Reserved	15	9
INT11.10	209	0x0000 0EA2	2	Reserved	15	10
INT11.11	210	0x0000 0EA4	2	Reserved	15	11
INT11.12	211	0x0000 0EA6	2	Reserved	15	12
INT11.13	212	0x0000 0EA8	2	Reserved	15	13
INT11.14	213	0x0000 0EAA	2	Reserved	15	14
INT11.15	214	0x0000 0EAC	2	Reserved	15	15
INT11.16	215	0x0000 0EAE	2	Reserved	15	16 (Lowest)
<b>PIE Group 12 Vectors - Muxed into CPU INT12</b>						
INT12.1	120	0x0000 0DF0	2	XINT3 interrupt	16	1 (Highest)
INT12.2	121	0x0000 0DF2	2	XINT4 interrupt	16	2
INT12.3	122	0x0000 0DF4	2	XINT5 interrupt	16	3
INT12.4	123	0x0000 0DF6	2	Reserved	16	4
INT12.5	124	0x0000 0DF8	2	Reserved	16	5
INT12.6	125	0x0000 0DFA	2	Reserved	16	6
INT12.7	126	0x0000 0DFC	2	FPU overflow interrupt	16	7
INT12.8	127	0x0000 0DFE	2	FPU underflow interrupt	16	8
INT12.9	216	0x0000 0EB0	2	Reserved	16	9
INT12.10	217	0x0000 0EB2	2	RAM correctable error interrupt	16	10
INT12.11	218	0x0000 0EB4	2	Flash correctable error interrupt	16	11
INT12.12	219	0x0000 0EB6	2	RAM access violation interrupt	16	12
INT12.13	220	0x0000 0EB8	2	PLL slip interrupt	16	13
INT12.14	221	0x0000 0EBA	2	Reserved	16	14

**Table 3-5. PIE Interrupt Vectors (continued)**

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
INT12.15	222	0x0000 0EBC	2	CLA overflow interrupt	16	15
INT12.16	223	0x0000 0EBE	2	CLA underflow interrupt	16	16 (Lowest)

## 3.6 Exceptions and Non-Maskable Interrupts

This section describes system-level error conditions that can trigger a non-maskable interrupt (NMI). The interrupt allows the application to respond to the error.

### 3.6.1 Configuring and Using NMIs

An incoming NMI sets a status bit in the NMIFLG register and starts the NMI watchdog counter. This counter is clocked by the SYSCLK, and if it reaches the value in the NMIWDPRD register, it triggers an NMI watchdog reset (NMIWDRS). To prevent this, the NMI handler must clear the flag bit using the NMIFLGCLR register. Once all flag bits are clear, the NMIINT bit in the NMIFLG register may also be cleared to allow future NMIs to be taken.

The NMI module is enabled by the boot ROM during the startup process. To respond to NMIs, an NMI handler vector must be written to the PIE vector table.

### 3.6.2 Emulation Considerations

The NMI watchdog counter behaves as follows under debug conditions:

CPU Suspended	When the CPU is suspended, the NMI watchdog counter will be suspended.
Run-Free Mode	When the CPU is placed in run-free mode, the NMI watchdog counter will resume operation as normal.
Real-Time Single-Step Mode	When the CPU is in real-time single-step mode, the NMI watchdog counter will be suspended. The counter remains suspended even within real-time interrupts.
Real-Time Run-Free Mode	When the CPU is in real-time run-free mode, the NMI watchdog counter operates as normal.

### 3.6.3 NMI Sources

There are several types of hardware errors that can trigger an NMI. Additional information about the error is usually available from the module that detects it.

#### 3.6.3.1 Missing Clock Detection

The missing clock detection logic monitors OSCCLK for failure. If the OSCCLK source stops, the PLL is bypassed, OSCCLK is connected to INTOSC1, and an NMI is fired to the CPU. For more information on missing clock detection, see [Section 3.7.12](#).

#### 3.6.3.2 RAM Uncorrectable ECC Error

A single-bit parity error, double-bit ECC data error, or single-bit ECC address error in a RAM read will trigger an NMI. This applies to CPU, CLA, and DMA reads. Single-bit ECC data errors do not trigger an NMI, but can optionally trigger a normal peripheral interrupt. For more information on RAM error detection, see [Section 3.11.1.7](#).

### 3.6.3.3 Flash Uncorrectable ECC Error

A double-bit ECC data error or single-bit ECC address error in a flash read will trigger an NMI. Single-bit ECC data errors do not trigger an NMI, but can optionally trigger a normal peripheral interrupt. For more information on flash error detection, see .

### 3.6.3.4 Software-Forced Error

There is a special NMI source that can only be triggered by writing to the SWERR bit in the NMIFLGFRM register. Since the SWERR flag is never set by a real hardware fail, it can be used to implement a self-test mode for the NMI subsystem.

### 3.6.4 Illegal Instruction Trap (ITRAP)

If the CPU tries to execute an illegal instruction, it generates a special interrupt called an illegal instruction trap (ITRAP). This interrupt is non-maskable and has its own vector in the PIE vector table. For more information about ITRAPs, see the Illegal-Instruction Trap section of the [TMS320C28x DSP CPU and Instruction Set Reference Guide](#).

---

**NOTE:** A RAM fetch access violation will trigger an ITRAP in addition to the normal peripheral interrupt for RAM access violations. The CPU will handle the ITRAP first.

---

### 3.6.5 Error Pin

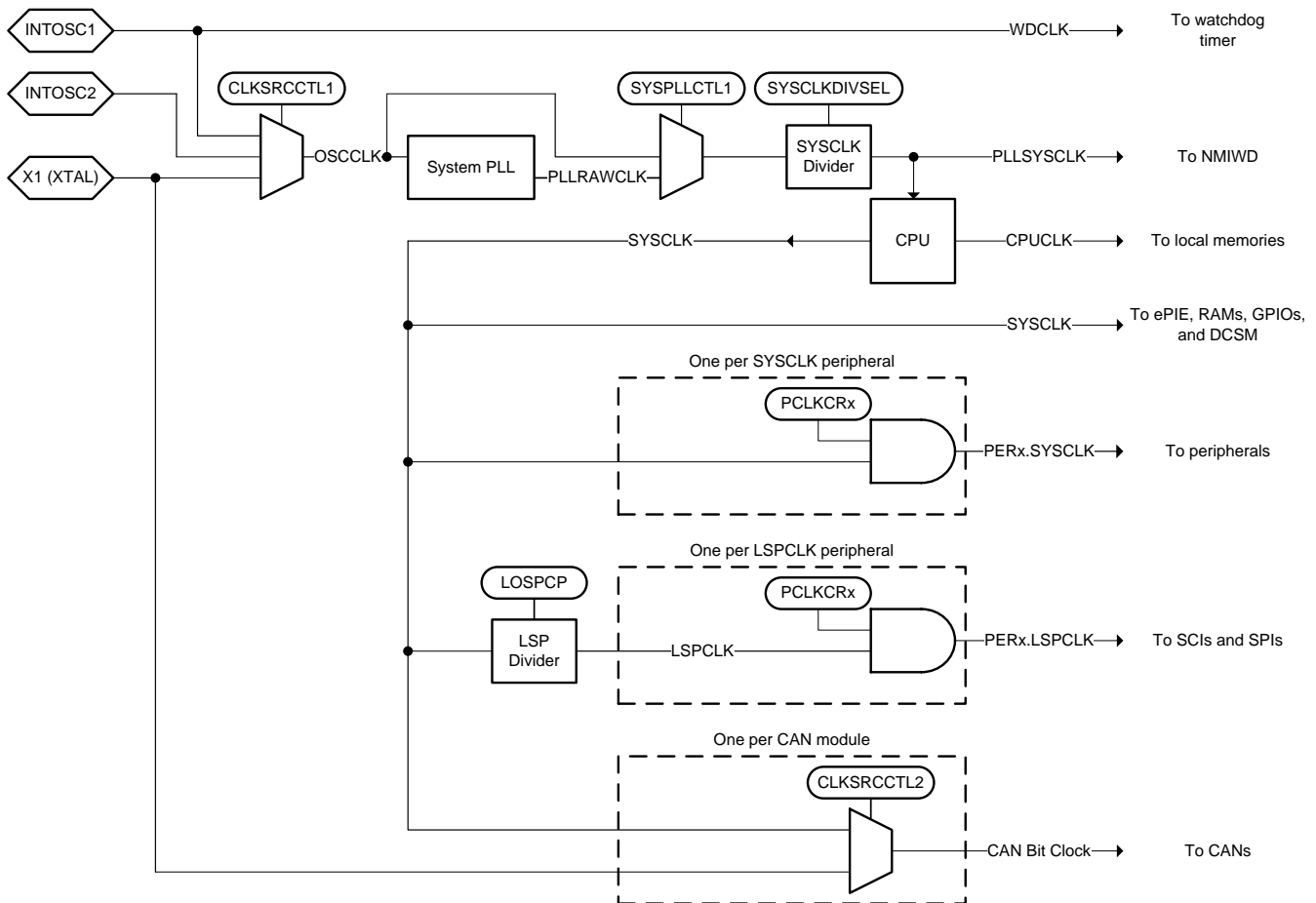
A signal called ERRORSTS can be output to GPIO24, GPIO28, or GPIO29. This signal goes low when any bit is set in the NMI shadow flag register (NMISHDFLG). It can be used to alert an external system to a problem in the microcontroller. Since the state of ERRORSTS is based on the shadow flags, ERRORSTS will remain low until the flags are cleared by the CPU or a power-on reset occurs.

All GPIO pins are inputs on power-up. If the state of the chosen ERRORSTS pin during power-up is important, an external pull-up should be connected to the pin.

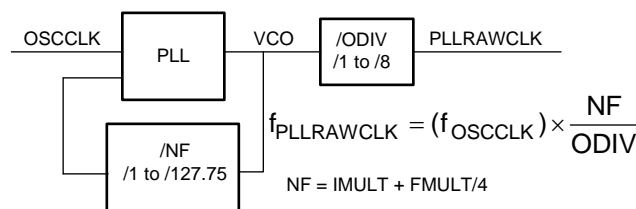
### 3.7 Clocking

This section explains the clock sources and clock domains on this device, and how to configure them for application use. [Figure 3-3](#) and [Figure 3-4](#) provide an overview of the device's clocking system.

**Figure 3-3. Clocking System**



**Figure 3-4. System PLL**



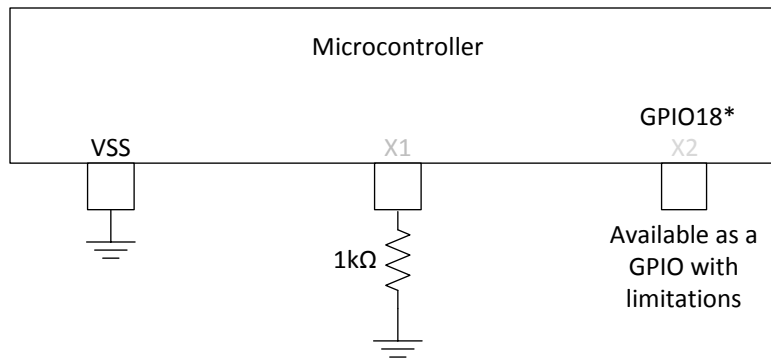
#### 3.7.1 Clock Sources

All of the clocks in the device are derived from one of four clock sources.

### 3.7.1.1 Primary Internal Oscillator (INTOSC2)

At power-up, the device is clocked from an on-chip 10 MHz oscillator (INTOSC2). INTOSC2 is the primary internal clock source, and is the default system clock at reset. It is used to run the boot ROM and can be used as the system clock source for the application. Note that INTOSC2's frequency tolerance is too loose to meet the timing requirements for CAN. Use of the CAN modules requires an external oscillator. When INTOSC2 is used as the system clock source, GPIO18 (X2) is available for use as a GPIO. A 1k pull-down resistor must be connected to X1. GPIO18 has different electrical characteristics from the other GPIOs due to interaction with the oscillator circuit. For more information, see the device datasheet.

**Figure 3-5. Using GPIO18 when INTOSC2 is the SYCLK source**



### 3.7.1.2 Backup Internal Oscillator (INTOSC1)

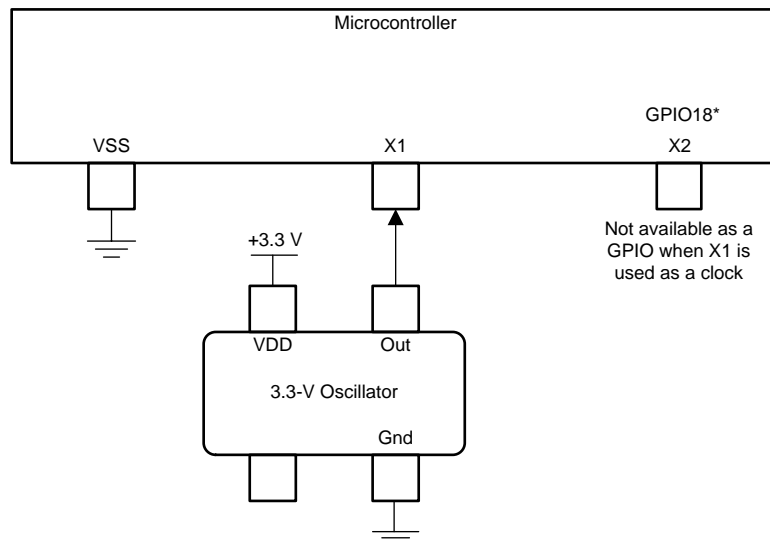
The device also includes a redundant on-chip 10 MHz oscillator (INTOSC1). INTOSC1 is a backup clock source that normally only clocks the watchdog timers and missing clock detection circuit (MCD). If MCD is enabled and a missing system clock is detected, the system PLL is bypassed and all system clocks are connected to INTOSC1 automatically. INTOSC1 may also be manually selected as the system clock source for debug purposes.

### 3.7.1.3 External Oscillator (XTAL)

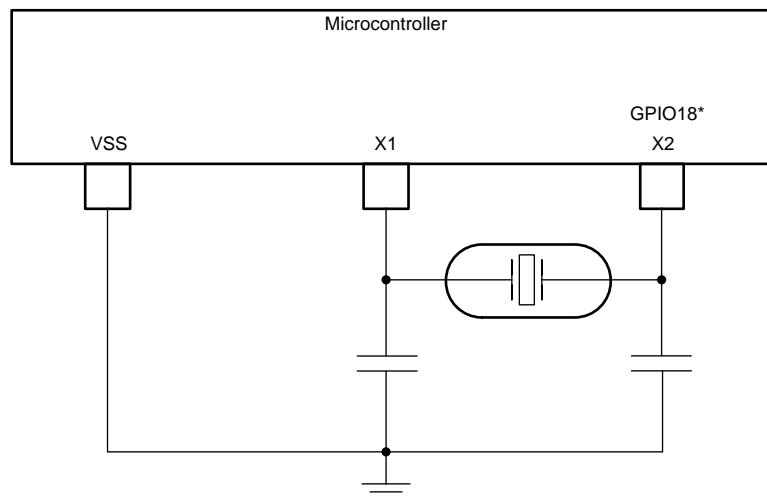
The device supports an external clock source (XTAL), which can be used as the main system and CAN bit clock source. Frequency limits and timing requirements can be found in the device datasheet. External clock sources use the X1 and X2/GPIO18 pins. After power-up, the X1 and X2 pin functionality can be enabled by following the procedure in [Section 3.7.6](#).

Three types of external clock sources are supported:

- A single-ended 3.3V external clock. The clock signal should be connected to X1, as shown in [Figure 3-6](#). X2/GPIO18 is not available as a GPIO.

**Figure 3-6. Single-ended 3.3V External Clock**


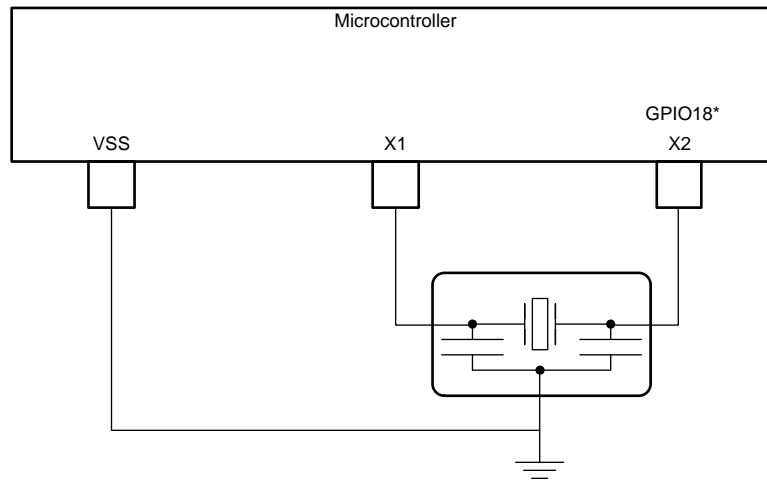
- An external crystal. The crystal should be connected across X1 and X2 with its load capacitors connected to VSS as shown in [Figure 3-7](#).

**Figure 3-7. External Crystal**


- An external resonator. The resonator should be connected across X1 and X2 with its ground connected to VSS as shown in [Figure 3-8](#).



**Figure 3-8. External Resonator**



### 3.7.2 Derived Clocks

The clock sources discussed in the previous section can be multiplied (via PLL) and divided down to produce the desired clock frequencies for the application. This process produces a set of derived clocks, which are described in this section.

#### 3.7.2.1 Oscillator Clock (OSCCLK)

One of INTOSC2, XTAL, or INTOSC1 must be chosen to be the master reference clock (OSCCLK) for the CPU and most of the peripherals. OSCCLK may be used directly or fed through the system PLL to reach a higher frequency. At reset, OSCCLK is the default system clock, and is connected to INTOSC2.

#### 3.7.2.2 System PLL Output Clock (PLLRAWCLK)

The system PLL allows the device to run at its maximum rated operating frequency, and in most applications will generate the main system clock. This PLL uses OSCCLK as a reference, and features a fractional multiplier. PLLRAWCLK is the output of the PLL's voltage-controlled oscillator (VCO). For configuration instructions, see [Section 3.7.6](#).

### 3.7.3 Device Clock Domains

The device clock domains feed the clock inputs of the various modules in the device. They are connected to the derived clocks, either directly or through an additional divider.

#### 3.7.3.1 System Clock (PLLSYSCLK)

The NMI watchdog timer has its own clock domain (PLLSYSCLK). Despite the name, PLLSYSCLK may be connected to the system PLL (PLLRAWCLK) or to OSCCLK. The chosen clock source is run through a frequency divider, which is configured via the SYSCLKDIVSEL register. PLLSYSCLK is gated in HALT mode.

#### 3.7.3.2 CPU Clock (CPUCLK)

The CPU has its own clock (CPUCLK) which is used to clock the CPU, its coprocessors, its private RAMs (M0 and M1), and the boot ROM and flash wrapper. This clock is identical to PLLSYSCLK, but is gated when the CPU enters IDLE or HALT mode.

### 3.7.3.3 CPU Subsystem Clock (SYSCLK and PERx.SYSCLK)

The CPU provides a clock (SYSCLK) to the CLA, DMA, and most peripherals. This clock is identical to PLLSYSCLK, but is gated when the CPU enters HALT mode.

Each peripheral clock has its own independent clock gating which is controlled by the PCLKCRx registers.

### 3.7.3.4 Low-Speed Peripheral Clock (LSPCLK and PERx.LSPCLK)

The SCI and SPI modules can communicate at bit rates that are much slower than the CPU frequency. These modules are connected to a shared clock divider, which generates a low-speed peripheral clock (LSPCLK) derived from SYSCLK. LSPCLK uses a /4 divider by default, but the ratio can be changed via the LOSPCP register. Each SCI and SPI module's clock (PERx.LSPCLK) can be gated independently via the PCLKCRx registers.

### 3.7.3.5 CAN Bit Clock

The required frequency tolerance for the CAN bit clock depends on the bit timing setup and network configuration, and can be as tight as 0.1%. Since the main system clock (in the form of PERx.SYSCLK) may not be precise enough, the bit clock can also be connected to XTAL via the CLKSRCCTL2 register. There is an independent selection for each CAN module.

To guarantee correct operation, the frequency of the CAN bit clock must be less than or equal to the SYSCLK frequency.

### 3.7.3.6 CPU Timer2 Clock (TIMER2CLK)

CPU timers 0 and 1 are connected to PERx.SYSCLK. Timer 2 is connected to PERx.SYSCLK by default, but may also be connected to INTOSC1, INTOSC2, or XTAL via the TMR2CLKCTL register. This register also provides a separate prescale divider for timer 2. If a non-SYSCLK source is used, it must be divided down to no more than half the SYSCLK frequency.

The main reason to use a non-SYSCLK source would be for internal frequency measurement. In most applications, timer 2 will run off of SYSCLK.

## 3.7.4 XCLKOUT

It is sometimes necessary to observe a clock directly for debug and testing purposes. The external clock output (XCLKOUT) feature supports this by connecting a clock to an external pin, which can be GPIO16 or GPIO18. The available clock sources are PLLSYSCLK, PLLRAWCLK, SYSCLK, INTOSC1, INTOSC2, and XTAL.

To use XCLKOUT, first select the clock source via the CLKSRCCTL3 register. Next, select the desired output divider via the XCLKOUTDIVSEL register. Finally, connect GPIO16 or GPIO18 to mux channel 11 using the GPIO configuration registers.

## 3.7.5 Clock Connectivity

The tables below provide details on the clock connections of every module present in the device.

**Table 3-6. Clock Connections Sorted by Clock Domain**

Clock Domain	Module Name
CPUCLK	CPU
	FPU
	TMU
	VCU
	Flash
	M0 - M1 RAMs
	Boot ROM
SYSCLK	ePIE

**Table 3-6. Clock Connections Sorted by Clock Domain (continued)**

<b>Clock Domain</b>	<b>Module Name</b>
	LS0 - LS7 RAMs GS0 - GS3 RAMs GPIO Input Sync and Qual CLA Message RAMs DCSM
PLLSYSCLK	NMIWD
PERx.SYSCLK	CLA Timer0 - 2 DMA ePWM1 - 8 eCAP1 - 7 eQEP1 - 2 SDFM1 - 4 ADCA - C CMPSS1 - 7 DACA - B PGA1 - 7 I2CA PMBUS LINA
PERx.LSPCLK	SCIA - B SPIA - B
CAN Bit Clock	CANA - B
WDCLK (INTOSC1)	Watchdog Timer

**Table 3-7. Clock Connections Sorted by Module Name**

Module Name	Clock Domain
ADCA - C	PERx.SYSCLK
Boot ROM	CPUCLK
CANA - B	CAN Bit Clock
CLA	PERx.SYSCLK
CLA Message RAMs	SYSCLK
CMPSS1 - 7	PERx.SYSCLK
CPU	CPUCLK
CPU Timers	PERx.SYSCLK
DACA - B	PERx.SYSCLK
DCSM	SYSCLK
DMA	PERx.SYSCLK
eCAP1 - 7	PERx.SYSCLK
ePIE	SYSCLK
ePWM1 - 8	PERx.SYSCLK
eQEP1 - 2	PERx.SYSCLK
Flash	CPUCLK
FPU	CPUCLK
GPIO Input Sync and Qual	SYSCLK
GS0 - GS3 RAMs	SYSCLK
I2CA	PERx.SYSCLK
LINA	PERx.SYSCLK
LS0 - LS7 RAMs	SYSCLK
M0 - M1 RAMs	CPUCLK
NMIWD	PLLSYSCLK
PGA1 - 7	PERx.SYSCLK
PMBUSA	PERx.SYSCLK
SCIA - B	PERx.LSPCLK
SDFM1 - 4	PERx.SYSCLK
SPIA - B	PERx.LSPCLK
TMU	CPUCLK
VCU	CPUCLK
Watchdog Timer	WDCLK (INTOSC1)

### 3.7.6 Clock Source and PLL Setup

The needs of the application are what ultimately determine the clock configuration. Specific concerns such as application performance, power consumption, total system cost, and EMC are beyond the scope of this document, but they should provide answers to the following questions:

1. What is the desired CPU frequency?
2. Is CAN required?
3. What types of external oscillators or clock sources are available?

If CAN is required, an external clock source with a precise frequency must be used as a reference clock. Otherwise, it may be possible to use only INTOSC2 and avoid the need for more external components.

### 3.7.7 Using an External Crystal or Resonator

The X1 and X2 pins double as GPIO19 and GPIO18. At power-up, these pins are in GPIO mode and the on-chip crystal oscillator is powered off. The following procedure can be used to switch the pins to X1 and X2 mode and enable the oscillator:

1. Clear the XTALCR.OSCOFF bit.
2. Clear the X1 counter by writing a 1 to X1CNT.CLR.
3. Wait for the X1 counter value in the X1CNT register to reach 1023 (0x3ff).
4. Select XTAL as the OSCCLK source by writing a 1 to CLKSRCCTL1.OSCCLKSRCSEL.
5. Check the MCLKSTS bit in the MCDPCR register. If it's set, the oscillator has not finished powering up, and more time is required:
  - a. Clear the missing clock status by writing a 1 to MCDPCR.MCLKCLR.
  - b. Repeat steps 2-5. Do not reset the device. Doing so will power down the oscillator, which requires the procedure to be restarted from step 1.
  - c. If the oscillator has not finished powering up in 10 milliseconds, there is a real clock failure.
6. If MCDPCR.MCLKSTS is clear, the oscillator startup is a success. The system clock is now derived from XTAL.

### 3.7.8 Using an External Oscillator

The procedure for using an external oscillator connected to the X1 pin is similar to the procedure for using a crystal or resonator:

1. Clear the XTALCR.OSCOFF bit.
2. Set the XTALCR.SE bit to enable single-ended mode.
3. Clear the X1 counter by writing a 1 to X1CNT.CLR.
4. Wait for the X1 counter value in the X1CNT register to reach 1023 (0x3ff).
5. Select XTAL as the OSCCLK source by writing a 1 to CLKSRCCTL1.OSCCLKSRCSEL.
6. Check the MCLKSTS bit in the MCDPCR register. If it's set, either the external oscillator or the device has failed.
7. If MCLKSTS is clear, the switch to the external clock is a success. The system clock is now derived from XTAL.

### 3.7.9 Choosing PLL Settings

There are two settings to configure for the PLL – a multiplier and a divider. They obey the formula:

$$f_{\text{PLLSYSCLK}} = f_{\text{OSCCLK}} * (\text{SYSPLLMULT.IMULT} + \text{SYSPLLMULT.FMULT}) / (\text{SYSPLLMULT.ODIV} * \text{SYSCLKDIVSEL.PLLSYSCLKDIV})$$

where  $f_{\text{OSCCLK}}$  is the system oscillator clock frequency, IMULT and FMULT are the integral and fractional parts of the multipliers, ODIV is the PLL output divider, and PLLSYSCLKDIV is the system clock divider. For the permissible values of the multipliers and dividers, see the documentation for their respective registers.

Many combinations of multiplier and divider can produce the same output frequency. However, the product of the reference clock frequency and the multiplier (known as the VCO frequency) must be in the range specified in the datasheet's  $f_{(\text{VCO})}$  parameter. The VCO frequency divided by the output divider (known as the PLL output frequency) must be in the range specified by the datasheet's  $f_{(\text{PLLRAWCLK})}$  parameter. The examples below will use 120 - 200 MHz for this parameter.

---

**NOTE:** The system clock frequency (PLLSYSCLK) may not exceed the limits specified in the data sheet's  $f_{(\text{SYSCLK})}$  parameter. These limits do not allow for oscillator tolerance.

---

### 3.7.10 System Clock Setup

Once the application requirements are understood, a specific clock configuration can be determined. The default configuration is for INTOSC2 to be used as the system clock (PLLSYSCLK) with a divider of 1. The following procedure can be used to set up the desired application configuration:

1. Select the reference clock source (OSCCLK) by writing to CLKSRCCTL1.OSCCLKSRCSEL. To enable XTAL, follow the instructions in the previous sections.

2. Set up the system PLL if desired. TI recommends using the C2000Ware SysCtl:setClock() function for proper configuration of the PLL clock.
3. Select the LSPCLK divider by writing to LOSPCP.
4. If an alternate CAN bit clock is needed, select it by writing to CLKSRCCTL2.CANABCLKSEL and CLKSRCCTL2.CANBBCLKSEL.
5. Enable the desired peripheral clocks by writing to the PCLKCRx registers.

The system clock configuration can be changed at run time. Changing the OSCCLK source will automatically bypass the PLL and set the multiplier to zero. Changing the multiplier from one non-zero value to another will temporarily bypass the PLL until it re-locks.

### 3.7.11 Clock Configuration Examples

**Example 1:** Using INTOSC2 (10 MHz) as a reference, generate a CPU frequency of 100 MHz - 3%:

```

CLKSRCCTL1.OSCCLKSRCSEL = 0x0
SYSPLLMULT = 0x00113
SYSCLKDIVSEL.PLLSYSCLKDIV = 2 (0x1)
SYSPLLCTL1.PLLCLKEN = 1
IMULT = 19 (0x13), FMULT = 0.25
(0x1), ODIV = 1 (0x0)
    
```

This gives a PLLRAWCLK of 192.5 MHz, which is within the acceptable range of 150 - 200 MHz. The CPU frequency is 96.25 MHz with a 3% tolerance due to variation in the internal oscillator.

**Example 2:** Using a crystal (20 MHz) as a reference, generate a CPU frequency of 85 MHz:

```

XTALCR.OSCOFF = 0
XTALCR.X1CNT.CLR = 1
(wait for X1CNT to reach 0x3ff)
CLKSRCCTL1.OSCCLKSRCSEL = 0x1
(check MDCR.MCLKSTS and repeat the above if
necessary)
SYSPLLMULT = 0x00208
SYSCLKDIVSEL.PLLSYSCLKDIV = 2 (0x1)
SYSPLLCTL1.PLLCLKEN = 1
IMULT = 8 (0x08), FMULT = .50
(0x2), ODIV = 1 (0x0)
    
```

This gives a PLLRAWCLK of 170 MHz, which is in the acceptable range. The CPU frequency is exactly 85 MHz.

---

#### NOTE:

1. SYSPLL must be bypassed and powered down manually before changing the OSCCLK source.
  2. At least 60 CPU clock cycles delay is needed after bypassing PLL i.e. SYSPLLCTL1.PLLCLKEN=0.
  3. At least 60 CPU clock cycles delay is needed after PLL is powered down i.e. SYSPLLCTL1.PLEN=0.
  4. At least 60 CPU clock cycles delay is needed after OSSCLK source is changed.
- 

### 3.7.12 Missing Clock Detection

The missing clock detection (MCD) subsystem detects OSCCLK failure using INTOSC1 as a reference clock. This subsystem only detects complete loss of OSCCLK. Frequency drift is not detected.

OSCCLK is connected to a 7-bit counter (MCDPCNT). INTOSC1 is connected to a 13-bit counter (MCDSCNT). When MCDPCNT overflows, MCDSCNT is reset. Thus, if OSCCLK is present and its frequency is greater than 1/64 of INTOSC1's frequency, MCDSCNT will never overflow. If OSCCLK stops for any reason, MCDSCNT will overflow and a missing clock condition will be detected.

When the MCD subsystem detects that OSCCLK is missing, the following occurs:

- The PLL is bypassed and OSCCLK is connected to INTOSC1 (after the PLLSYSCLK divider). PLLMULT is zeroed out automatically.
- The MCDCR.MCDSTS flag is set. While this flag is set, the OSCCLKSRCSEL bits have no effect.
- The CLOCKFAIL signal goes high, which generates trip events to the PWM modules and triggers an NMI.
- The MCDSCNT counter is frozen to prevent further missing clock detection.

To clear a missing clock condition, write a 1 to the MCDCR.MCLKCLR bit. This will restore the functionality of the OSCCLKSRCSEL bits and reset the MCD counters. To lock the PLL during a missing clock condition, switch the OSCCLK source to INTOSC1 using the OSCCLKSRCSEL bits, clear the missing clock condition, then write to the PLL registers.

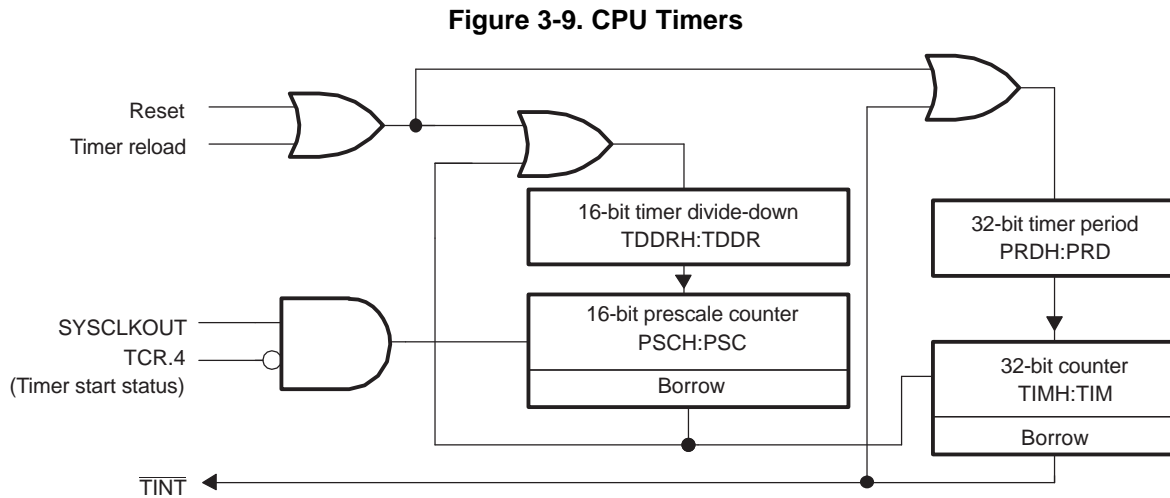
Missing clock detection is enabled at startup.



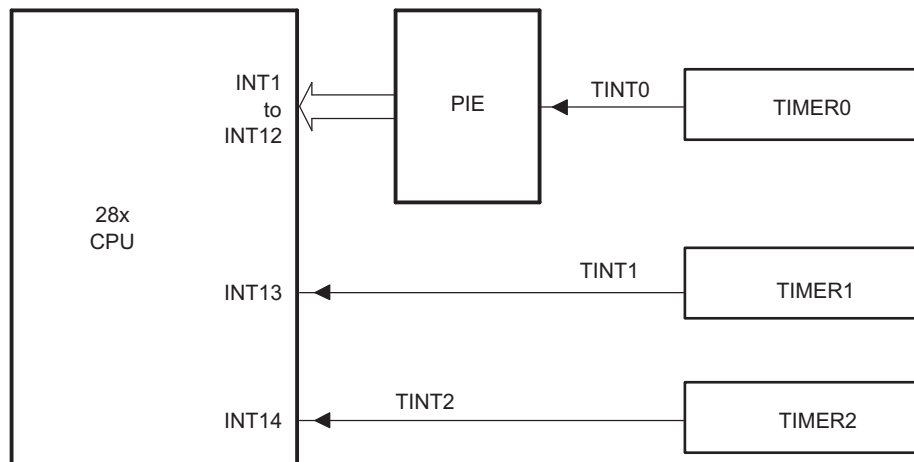
### 3.8 32-Bit CPU Timers 0/1/2

This section describes the three 32-bit CPU timers (TIMER0/1/2) shown in [Figure 3-9](#).

Timer0 and Timer1 can be used in user applications. Timer2 is reserved for real-time operating system uses (for example, TI-RTOS). If the application is not using an operating system that utilizes this timer, then Timer2 can be used in the application. timer interrupt signals (TINT0, TINT1, TINT2) are connected as shown in [Figure 3-10](#).



**Figure 3-10. CPU Timer Interrupt Signals and Output Signal**



- A The timer registers are connected to the memory bus of the C28x processor.
- B The CPU Timers are synchronized to SYSCLKOUT.

The general operation of a CPU timer is as follows:

- The 32-bit counter register, TIMH:TIM, is loaded with the value in the period register PRDH:PRD
- The counter decrements once every  $(TPR[TDDR:H:TDDR]+1)$  SYSCLK cycles, where TDDR:H:TDDR is the timer divider.
- When the counter reaches 0, a timer interrupt output signal generates an interrupt pulse.

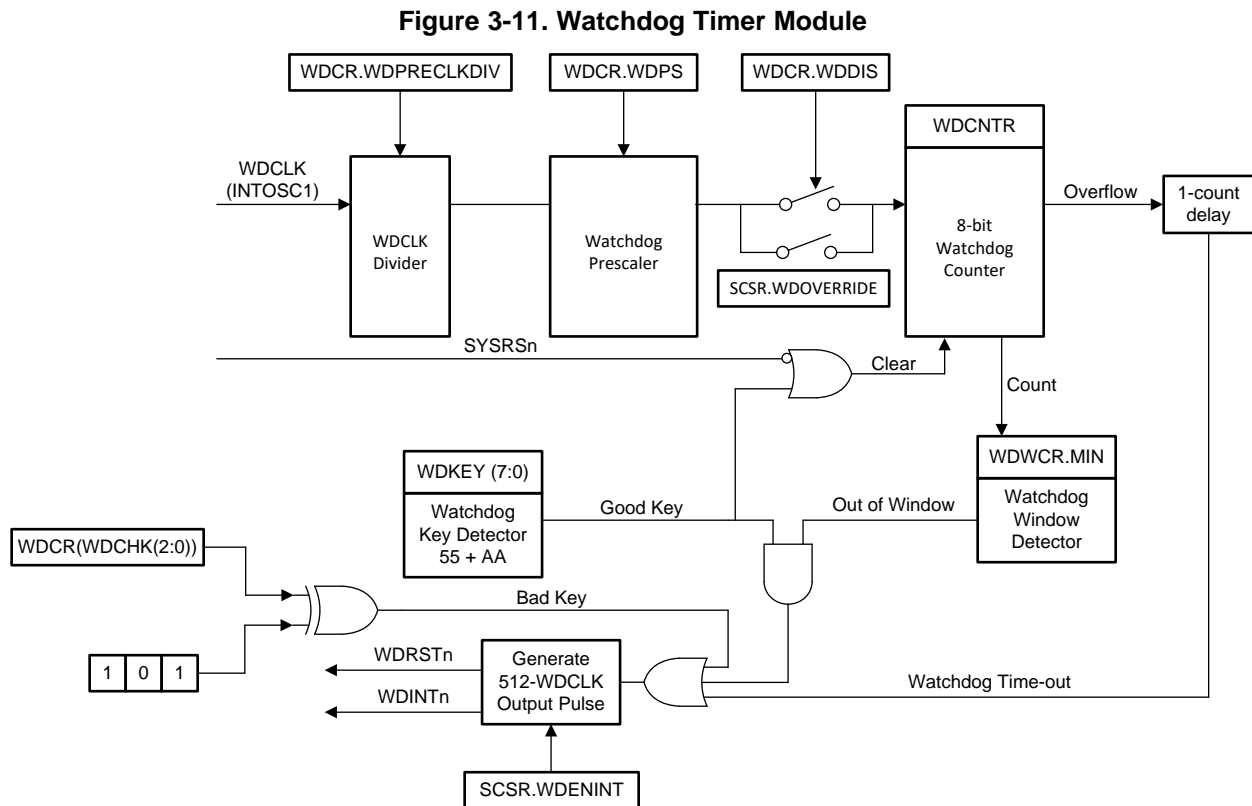
The registers listed in [Section 3.14](#) are used to configure the timers.

### 3.9 Watchdog Timer

The watchdog module consists of an 8-bit counter fed by a prescaled clock (WDCLK, which is connected to INTOSC1). When the counter reaches its maximum value, the module generates an output pulse 512 WDCLKs wide. This pulse can generate an interrupt or a reset. The CPU must periodically write a 0x55 + 0xAA sequence into the watchdog key register to reset the watchdog counter. The counter can also be disabled.

The counter's clock is divided down from WDCLK by two dividers. The prescaler is adjustable from /1 to /64 in powers of two. The pre-divider defaults to /512 for backwards compatibility, but is adjustable from /2 to /4096 in powers of two. This allows a wide range of timeout values for safety-critical applications.

Figure 3-11 shows the various functional blocks within the watchdog module.



#### 3.9.1 Servicing the Watchdog Timer

The watchdog counter (WDCNTR) is reset when the proper sequence is written to the WDKEY register before the 8-bit watchdog counter overflows. The WDCNTR is reset-enabled when a value of 0x55 is written to the WDKEY. When the next value written to the WDKEY register is 0xAA, then the WDCNTR is reset. Any value written to the WDKEY other than 0x55 or 0xAA causes no action. Any sequence of 0x55 and 0xAA values can be written to the WDKEY without causing a system reset; only a write of 0x55 followed by a write of 0xAA to the WDKEY resets the WDCNTR.

**Table 3-8. Example Watchdog Key Sequences**

Step	Value Written to WDKEY	Result
1	0xAA	No action
2	0xAA	No action
3	0x55	WDCNTR is enabled to be reset if next value is 0xAA.
4	0x55	WDCNTR is enabled to be reset if next value is 0xAA.
5	0x55	WDCNTR is enabled to be reset if next value is 0xAA.
6	0xAA	WDCNTR is reset.
7	0xAA	No action
8	0x55	WDCNTR is enabled to be reset if next value is 0xAA.
9	0xAA	WDCNTR is reset.
10	0x55	WDCNTR is enabled to be reset if next value is 0xAA.
11	0x32	Improper value written to WDKEY. No action, WDCNTR no longer enabled to be reset by next 0xAA.
12	0xAA	No action due to previous invalid value.
13	0x55	WDCNTR is enabled to be reset if next value is 0xAA.
14	0xAA	WDCNTR is reset.

The first action that enables the WDCNTR to be reset is shown in Step 3 in [Table 3-8](#). The WDCNTR is not actually reset until step 6. Step 8 again re-enables the WDCNTR to be reset and step 9 resets the WDCNTR. Step 10 again re-enables the WDCNTR to be reset. Writing the wrong key value to the WDKEY in step 11 causes no action, however the WDCNTR is no longer enabled to be reset and the 0xAA in step 12 now has no effect.

If the watchdog is configured to reset the device, then a WDCR overflow or writing the incorrect value to the WDCR[WDCHK] bits will reset the device and set the watchdog flag (WDRSn) in the reset cause register (RESC). After a reset, the program can read the state of this flag to determine whether the reset was caused by the watchdog. After doing this, the program should clear WDRSn to allow subsequent watchdog resets to be detected. Watchdog resets are not prevented when the flag is set.

### 3.9.2 Minimum Window Check

To complement the timeout mechanism, the watchdog also contains an optional "windowing" feature that requires a minimum delay between counter resets. This can help protect against error conditions that bypass large parts of the normal program flow but still include watchdog handling.

To set the window minimum, write the desired minimum watchdog count to the WDWCR register. This value will take effect after the next WDKEY sequence. From then on, any attempt to service the watchdog when WDCNTR is less than WDWCR will trigger a watchdog interrupt or reset. When WDCNTR is greater than or equal to WDWCR, the watchdog can be serviced normally.

At reset, the window minimum is zero, which disables the windowing feature.

### 3.9.3 Watchdog Reset or Watchdog Interrupt Mode

The watchdog can be configured in the SCSR register to either reset the device ( $\overline{\text{WDRST}}$ ) or assert an interrupt ( $\overline{\text{WDINT}}$ ) if the watchdog counter reaches its maximum value. The behavior of each condition is described below:

- **Reset mode:**

If the watchdog is configured to reset the device, then the  $\overline{\text{WDRST}}$  signal will pull the device reset ( $\overline{\text{XRS}}$ ) pin low for 512 OSCCLK cycles when the watchdog counter reaches its maximum value.

- **Interrupt mode:**

When the watchdog counter expires, it will assert an interrupt by driving the  $\overline{\text{WDINT}}$  signal low for 512 OSCCLK cycles. The falling edge of  $\overline{\text{WDINT}}$  triggers a WAKEINT interrupt in the PIE if it is enabled. Because the PIE is edge-triggered, re-enabling the WAKEINT while  $\overline{\text{WDINT}}$  is active will not produce a duplicate interrupt.

To avoid unexpected behavior, software should not change the configuration of the watchdog while  $\overline{\text{WDINT}}$  is active. For example, changing from interrupt mode to reset mode while  $\overline{\text{WDINT}}$  is active will immediately reset the device. Disabling the watchdog while  $\overline{\text{WDINT}}$  is active will cause a duplicate interrupt if the watchdog is later re-enabled. If a debug reset is issued while  $\overline{\text{WDINT}}$  is active, the reset cause register (RESC) will show a watchdog reset. The WDINTS bit in the SCSR register can be read to determine the current state of  $\overline{\text{WDINT}}$ .

### 3.9.4 Watchdog Operation in Low Power Modes

In IDLE mode, the watchdog interrupt ( $\overline{\text{WDINT}}$ ) signal can generate an interrupt to the CPU to take the CPU out of IDLE mode. As with any other peripheral, the watchdog interrupt will trigger a WAKE interrupt in the PIE during IDLE mode. User software must determine which peripheral caused the interrupt.

**Note:** If the watchdog interrupt is used to wake-up from an IDLE low power mode condition, software must make sure that the  $\overline{\text{WDINT}}$  signal goes back high before attempting to reenter the IDLE mode. The  $\overline{\text{WDINT}}$  signal will be held low for 512 OSCCLK cycles when the watchdog interrupt is generated. The current state of  $\overline{\text{WDINT}}$  can be determined by reading the watchdog interrupt status bit (WDINTS) bit in the SCSR register. WDINTS follows the state of  $\overline{\text{WDINT}}$  by two SYSCLKOUT cycles.

In HALT mode, the internal oscillators and watchdog timer are kept active if the user sets `CLKSRCCTL1.WDHALTI = 1`. A watchdog reset can wake the system from HALT mode, but a watchdog interrupt cannot.

### 3.9.5 Emulation Considerations

The watchdog module behaves as follows under various debug conditions:

CPU Suspended:	When the CPU is suspended, the watchdog clock (WDCLK) is suspended
Run-Free Mode:	When the CPU is placed in run-free mode, then the watchdog module resumes operation as normal.
Real-Time Single-Step Mode:	When the CPU is in real-time single-step mode, the watchdog clock (WDCLK) is suspended. The watchdog remains suspended even within real-time interrupts.
Real-Time Run-Free Mode:	When the CPU is in real-time run-free mode, the watchdog operates as normal.

### 3.10 Low Power Modes

All low-power modes are entered by setting the LPMCR register and executing the IDLE instruction. More information about this instruction can be found in the [TMS320C28x CPU and Instruction Set Reference Guide](#).

Low-power modes should not be entered into while a flash program or erase operation is ongoing. Entering HALT will stop all CPU and peripheral activities. This includes active transmissions and control algorithms. When preparing to enter HALT mode, the application should ensure that the system is prepared to enter a period of inactivity.

Before entering HALT mode check the value of the GPIODAT register of the pin selected for HALT wake-up (GPIOLPMSELO/1) prior to entering the low-power mode to ensure that the wake event has not already been asserted.

#### 3.10.1 IDLE

IDLE is a standard feature of the C28x CPU. In this mode, the CPU clock is gated while all peripheral clocks are left running. IDLE can thus be used to conserve power while a CPU is waiting for peripheral events.

Any enabled interrupt will wake the CPU up from IDLE mode.

To enter IDLE mode, set LPMCR.LPM to 0x0 and execute the IDLE instruction.

The CPU will resume normal operations upon any enabled interrupt event.

#### 3.10.2 Guidelines on Software Emulation of STANDBY Mode

STANDBY is not supported on this device, but can be emulated using the steps below in order to conserve power:

1. Switch SYSCLK source from PLLCLK to OSCCLK by configuring SYSPLLCTL1.PLLCLKEN=0
2. Disable peripheral clocks through PCLKCRx registers
3. Enter IDLE mode using the instructions defined in [Section 3.10.1](#).

Here are the key differences between Software Emulated STANDBY vs STANDBY mode.

**Table 3-9. Software Emulated STANDBY vs STANDBY Mode**

Software Emulated STANDBY	STANDBY (Not Supported)
Peripheral clocks needs to be disabled prior to entering IDLE mode.	Peripheral clocks are automatically disabled.
Wakeup source is XINT or WDINT.	Wakeup source is WAKEINT or WDINT.
Peripheral Interrupts should be disabled prior to entering IDLE mode, since any interrupt going into the PIE will wake up the system.	System can only wake up externally through WAKENIT (dedicated LPM Interrupt).
Any GPIO can be routed through INPUT XBAR to XINT as a wake up source. Qualification of the GPIO is through GPxQSEL.	GPIO has to be configured as LPM interrupt through GPIOLPMSELx to trigger WAKEINT. Qualification of the GPIO is through dedicated Low Power Mode register.
On wakeup, the CPU enters XINT ISR.	On wakeup, the CPU enters WAKEINT ISR.
Any peripheral clocks and Interrupts disabled prior to entering IDLE mode should be re-enabled by software on Wakeup.	System resumes normal operation on wake up. No extra steps needed.
If SYSCLK is switched from PLLCLK to OSCCLK prior to entering IDLE mode, it has to be switched back by configuring SYSPLLCTL1.PLLCLKEN=1 to resume normal operation	System resumes normal operation on wake up. No extra steps needed.

### 3.10.3 HALT

HALT is a global low-power mode that gates almost all system clocks and allows for power-down of oscillators and analog blocks.

Unlike on other C2000 devices, HALT mode will not automatically power down the XTAL upon HALT entry. Additionally, if the XTAL is not powered on, waking up from HALT mode will not automatically power on the XTAL. The XTALCR.OSCOFF bit has been added to power on and off the XTAL circuitry when not needed through application software.

For applications that require minimal power consumption during HALT mode, application software should power off the XTAL prior to entering HALT. If the OSCCLK source is configured to be XTAL, the application should first switch the OSSCLK source to INTOSC1 or INTOSC2 prior to setting XTALCR.OSCOFF.

GPIO0-63 can be configured to wake up the system from HALT. No other wakeup option is available. However, the watchdog timer may still be clocked, and can be configured to produce a watchdog reset if a timeout mechanism is needed. On wakeup, the CPU receives a WAKEINT interrupt.

#### To enter HALT mode:

1. Enable the WAKEINT interrupt in the PIE .
2. Set LPMCR.LPM to 0x2. Set GPIOLPMSEL0 and GPIOLPMSEL1 to connect the chosen GPIOs to the LPM module.
3. Set CLKSRCCTL1.WDHALTI to 1 to keep the watchdog timer active and INTOSC1 and INTOSC2 powered up in HALT.
4. Set CLKSRCCTL1.WDHALTI to 0 to disable the watchdog timer and power down INTOSC1 and INTOSC2 in HALT.
5. Execute the IDLE instruction to enter HALT.

If an interrupt or NMI is received while the IDLE instruction is in the pipeline, the system will begin executing the WAKEINT ISR. After HALT wakeup, ISR execution will resume where it left off.

**NOTE:** Before entering HALT mode, if the system PLL is locked (SYSPLL.LOCKS = 1), it must also be connected to the system clock (PLLCTL1.PLLCLKEN = 1). Otherwise, the device will never wake up.

#### To wake up from HALT mode:

1. Drive the selected GPIO low for a minimum 5us. This will activate the WAKEINT PIE interrupt.
2. Drive the wake-up GPIO high again to initiate the powering up of the SYSPLL
3. Wait 16us plus 1024 OSCLK cycles to allow the PLLs to lock and the WAKEINT ISR to be latched.
4. Execute the WAKEINT ISR.

The device is now out of HALT mode and can resume normal execution.

### 3.10.4 Flash Power-down Considerations

The Flash module on this device can be powered down at any time during an application. There are some considerations that must be made when powering down the Flash.

When the application software powers down the Flash, it must ensure that the function that puts the Flash to sleep is executed from RAM. Note that there should not be any access to Flash after the Flash is put to sleep to realize the power savings. If there is an access to Flash, Flash wakeup process (wakeup time depends on PSLEEP and RWAIT as mentioned in [Flash/OTP and Pump Power Modes and Wakeup](#) gets initialized and the application will not realize Flash power savings. For example, if the application has to execute any code after putting the Flash to sleep and before putting the device in to low power mode, the application should execute that code from RAM and not from Flash.

As mentioned in [Flash/OTP and Pump Power Modes and Wakeup](#), PSLEEP and RWAIT can be optimized to reduce the Flash wakeup time for a given SYSCLK frequency. BootROM configures the best possible PSLEEP value for the 100MHz operation. However, the application software can decrease the PSLEEP value to reduce the Flash wakeup time if the application SYSCLK is less than 100MHz. This is applicable in the context of an application entering the Halt mode since PLL must be disabled before entering the Halt mode. In this case, the PSLEEP value can be decreased to get a faster Flash wakeup upon exit from LPM.

If the Wake ISR is in Flash, it is suggested to optimize the PSLEEP and RWAIT values before entering the LPM, and after the Flash is in sleep, since application does not get a chance to modify these before Flash is awake after exiting from LPM. However, after the LPM exit and once the Flash is awake, application should branch to RAM to restore RWAIT and PSLEEP (as per the application SYSCLK to which PLL will be locked for) and then proceed with Flash execution to lock the PLL.

If the Wake ISR is in RAM, application can optimize the PSLEEP and RWAIT values in the Wake ISR and then do a dummy Flash access to initiate the Flash wakeup process. While the Flash is waking up, application can initialize the PLL lock process. Once the Flash is awake, application can put the PLL in clock path. If the user does not want to lock the PLL from RAM, PLL can be locked from Flash (this means Flash wakeup and PLL lock are not done in parallel), but in any case make sure to restore the RWAIT and PSLEEP (as per the application SYSCLK to which PLL will be locked for) in Wake ISR before proceeding to Flash execution.

Unlike F28M3x, F2837x, and F2807x devices, the Flash fallback mode is not configured as active mode automatically upon Flash wakeup in F28004x and instead stays as it is configured before entering the Flash low power mode. Hence, the BootROM code and the Flash initialization routine in C2000Ware configure the Flash fallback mode to active mode to avoid Flash falling back to low power mode after the grace period expiration. Similarly, in the context of the device LPM, the application software must change the Flash Fallback mode to the Active state in the Wake ISR if required. If not, Flash will enter the configured fallback power mode when the grace period expires as mentioned in [Flash/OTP and Pump Power Modes and Wakeup](#). This applies to all low power modes on this device.



### 3.11 Memory Controller Module

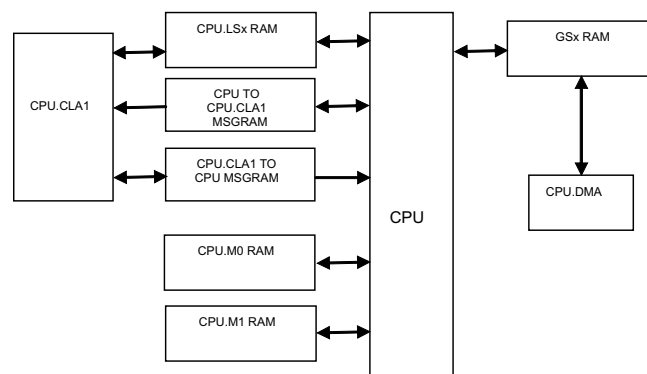
For these devices, the RAMs have different characteristics. Some are:

- dedicated to the CPU (M0, and M1),
- shared between the CPU and CLA (LSx RAM),
- shared between the CPU and DMA (GSx RAM), and
- used to send and receive messages between processors (MSGRAM).

All these RAMs are highly configurable to achieve control for write access and fetch access from different masters. All dedicated RAMs are enabled with the ECC feature (both data and address) and shared RAMs, are enabled with the parity (both data and address) feature. Some of the dedicated memories are secure memory as well. Refer to [Section 3.13](#) for more details. Each RAM has its own controller which takes care of the access protection/security related checks and ECC/Parity features for that RAM.

[Figure 3-12](#) shows the configuration of these RAMs.

**Figure 3-12. Memory Architecture**



**NOTE:** All RAMs on these devices are SRAMs.

#### 3.11.1 Functional Description

This section further defines and discusses the dedicated RAMs, shared RAMs, and MSG RAMs on this device.

##### 3.11.1.1 Dedicated RAM (Mx RAM)

This device has two dedicated RAM blocks: M0 and M1. M0 and M1 memories are small blocks of memory which are tightly coupled with the CPU. Only the CPU has access to these memories. No other masters (including DMA) have any access to these memories.

All dedicated RAMs have the ECC feature.

##### 3.11.1.2 Local Shared RAM (LSx RAM)

RAM blocks which are accessible to the CPU and CLA only, are called local shared RAMs (LSx RAMs). All such memories are secure memory and have the parity feature. By default, these memories are dedicated to the CPU only, and the user could choose to share these memories with the CLA by appropriately configuring the MSEL\_LSx bit field in the LSxMSEL register. Further, when these memories are shared between the CPU and CLA, the user could choose to use these memories as CLA program memory by configuring the CLAPGM\_LSx bit field in the LSxCLAPGM registers. CPU access to all memory blocks which are programmed as CLA program memory are blocked.

All these RAMs have the access protection (CPU write/CPU fetch) feature. Each type of access protection for each RAM block can be enabled or disabled by configuring the specific bit in the local shared RAM access protection registers. [Table 3-10](#) shows the LSx RAM features.

**Table 3-10. Local Shared RAM**

MSEL_LSx	CLAPGM_LSx	CPUx Allowed Access	CPUx.CLA1 Allowed Access	Comment
00	X	All	-	LSx memory is configured as CPU dedicated RAM
01	0	All	Data Read Data Write Emulation Data Read Emulation Data Write	LSx memory is shared between CPU and CLA1
01	1	Emulation Read Emulation Write	Fetch Only Emulation Program Read Emulation Program Write	LSx memory is CLA1 program memory

### 3.11.1.3 Global Shared RAM (GSx RAM)

RAM blocks which are accessible from the CPU and DMA are called global shared RAMs (GSx RAMs). [Table 3-11](#) shows the features of the GSx RAM.

**Table 3-11. Global Shared RAM**

CPU (Fetch)	CPU (Read)	CPU (Write)	CPU.DMA (Read)	CPU.DMA (Write)
Yes	Yes	Yes	Yes	Yes

Like other shared RAM, these RAMs also have different levels of access protection which can be enabled or disabled by configuring specific bits in the GSxACCPROT registers.

Master select and access protection configuration for each GSx RAM block can be individually locked by the user to prevent further update to these bit fields. The user can also choose to permanently lock the configuration to individual bit fields by setting the specific bit fields in the GSxCOMMIT register (refer to the register description for more details). Once a configuration is committed for a particular GSx RAM block, it can not be changed further until CPU.SYSRS is issued.

### 3.11.1.4 Message RAM (CLA MSGRAM)

These RAM blocks are used to share data between the CPU and CLA. The CLA has read and write access to the CLA to CPU MSGRAM. The CPU has read and write access to the "CPU to CLA MSGRAM." The CPU and CLA both have read access to both MSGRAMs.

This RAM has parity.

### 3.11.1.5 Access Arbitration

For a shared RAM, multiple accesses can happen at a given time. The maximum number of accesses to any shared RAM at any given time depends on the type of shared RAM. On this device, a combination of a fixed and round robin scheme is followed to arbitrate multiple access at any given time.

The following is the order of fixed priority for CPU accesses:

1. Data Write/Program Write
2. Data Read
3. Program Read/Program Fetch

The following is the order of fixed priority for CLA accesses:

1. Data Write
2. Data Read/Program Fetch

[Figure 3-13](#) represents the arbitration scheme on global shared memories:

**Figure 3-13. Arbitration Scheme on Global Shared Memories**

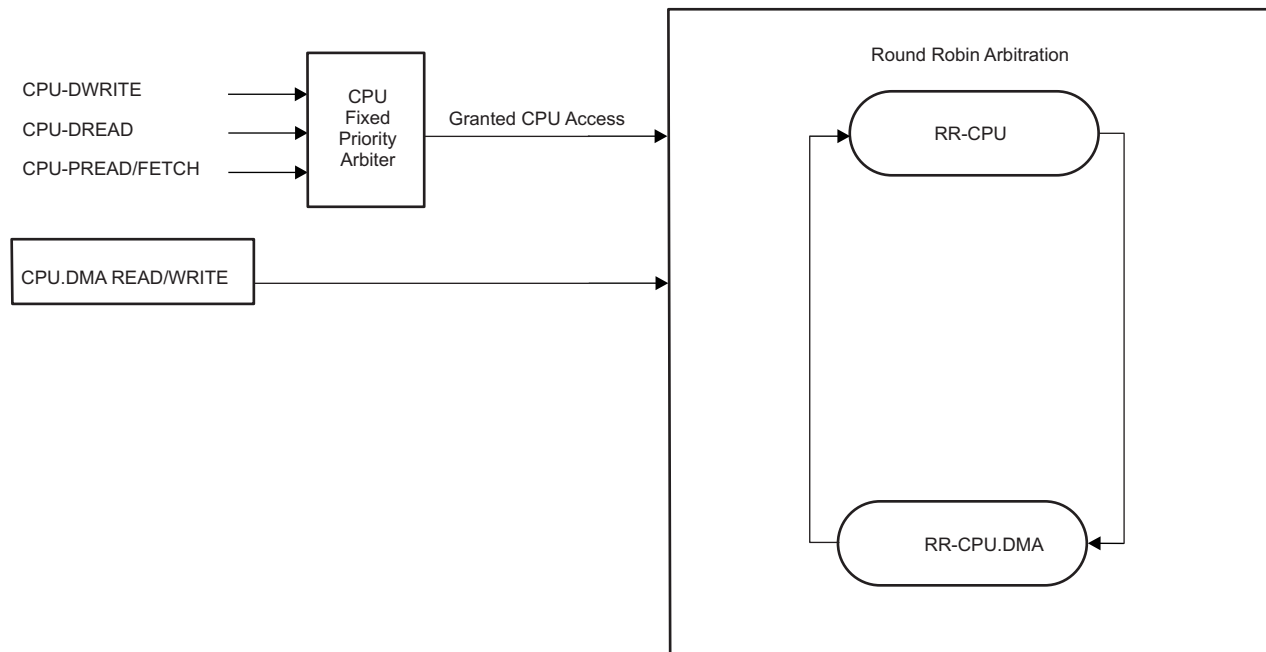
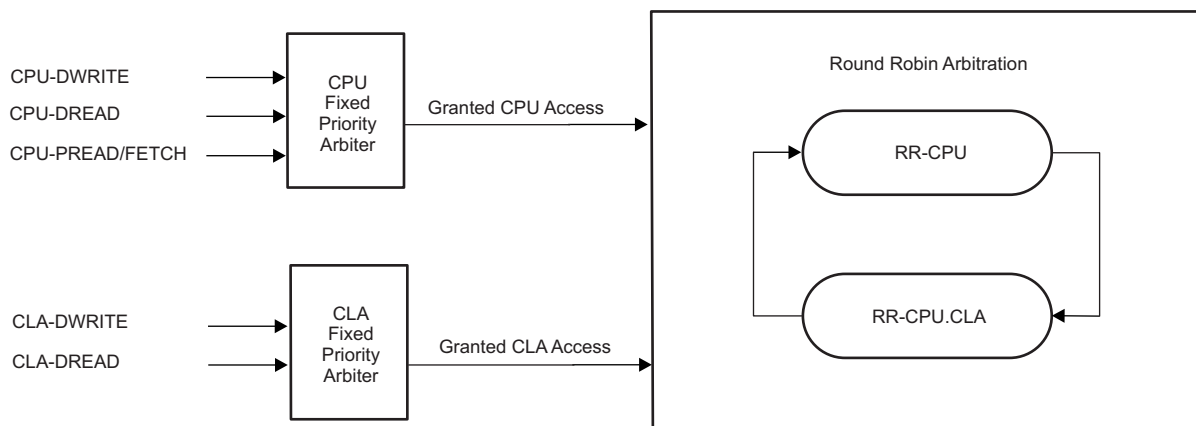


Figure 3-14 represents arbitration scheme on local shared memories.

**Figure 3-14. Arbitration Scheme on Local Shared Memories**



### 3.11.1.6 Access Protection

All RAM blocks except for M0/M1 have different levels of protection. This feature allows the user to enable or disable specific access to individual RAM blocks from individual masters. There is no protection for read accesses, hence reads are always allowed from all the masters which have access to that RAM block.

The following sections describe the different kinds of protection available for RAM blocks on this device.

---

**NOTE:** For debug accesses, all the protections are disabled.

---

#### 3.11.1.6.1 CPU Fetch Protection

Fetch accesses from the CPU can be protected by setting the FETCHPROTx bit of the specific register to '1.' If fetch access is done by the CPU to a memory where CPU fetch protection is enabled, a fetch protection violation occurs.

If a fetch protection violation occurs, it results in an ITRAP for CPU. A flag gets set in the appropriate access violation flag register, and the memory address for which the access violation occurred, gets latched into the appropriate CPU fetch access violation address register.

#### **3.11.1.6.2 CPU Write Protection**

Write accesses from the CPU can be protected by setting the CPUWRPROTx bit of the specific register to '1.' If write access is done by a CPU to memory where it is protected, a write protection violation occurs.

If a write protection violation occurs, the write gets ignored, a flag gets set in the appropriate access violation flag register, and the memory address for which the access violation occurred, gets latched into the appropriate CPU write access violation address register. Also, an access violation interrupt is generated if enabled in the interrupt enable register.

#### **3.11.1.6.3 CPU Read Protection**

For local shared RAM, if memory is shared between the CPU and its CLA, the CPU will only have access if the memory is configured as data RAM for the CLA. If it is programmed as program RAM, all the accesses from the CPU, including a read, will be blocked and the violation will be considered a non-master access violation.

If a read protection violation occurs, a flag gets set in the appropriate access violation flag register, and the memory address for which the access violation occurred, gets latched in the appropriate CPU read access violation address register. Also, an access violation interrupt is generated, if enabled in the interrupt enable register.

#### **3.11.1.6.4 CLA Fetch Protection**

If local shared RAM is configured as dedicated RAM for the CPU, or if it is configured as data RAM for the CLA, any fetch access from the CLA to that particular LSx RAM results in a CLA fetch protection violation, which is a non-master access violation.

If a CLA fetch protection violation occurs, it results in a MSTOP. A flag gets set in the appropriate access violation flag register, and the memory address for which the access violation occurred, gets latched in the appropriate CLA fetch access violation address register. Also, an access violation interrupt is generated to the master CPU if enabled in the interrupt enable register.

#### **3.11.1.6.5 CLA Write Protection**

If local shared RAM is configured as dedicated RAM for the CPU, or if it is configured as program RAM for the CLA, any data write access from the CLA to that particular LSx RAM results in a CLA write protection violation, which is a non-master access violation.

If a CLA write protection violation occurs, the write gets ignored, a flag gets set in the appropriate access violation flag register, and the memory address for which the access violation occurred, gets latched in the appropriate CLA write access violation address register. Also, an access violation interrupt is generated to the CPU if enabled in the interrupt enable register.

#### **3.11.1.6.6 CLA Read Protection**

If local shared RAM is configured as dedicated RAM for the CPU, or if it is configured as program RAM for the CLA, any data read access from the CLA to that particular LSx RAM results in a CLA read protection violation, which is a non-master access violation.

If a CLA read protection violation occurs, a flag gets set in the appropriate access violation flag register, and the memory address for which the access violation occurred, gets latched in the appropriate CLA read access violation address register. Also, an access violation interrupt is generated to the master CPU if enabled in the interrupt enable register.

#### **3.11.1.6.7 DMA Write Protection**

Write accesses from the DMA can be protected by setting the DMAWRPROTx bit of a specific register to '1.' If a write access is done by the DMA to protected memory, a write protection violation occurs.

If a write access is made to GSx memory by a non-master DMA, it is called a non-master write protection violation. If a write access is made to a dedicated or shared memory by a master DMA, and DMAWRPROTx is set to '1' for that memory, it is called a master DMA write protection violation.

A flag gets set in the DMA access violation flag register, and the memory address where the violation happened gets latched in the DMA fetch access violation address register. These are dedicated registers for each subsystem.

- Note 1:** All access protections are ignored during debug accesses. Write access to a protected memory will go through when it is done via the debugger, irrespective of the write protection configuration for that memory.
- Note 2:** Access protection is not implemented for M0 and M1 memories.
- Note 3:** In the case of local shared RAM, if memory is shared between the CPU and its CLA, the CPU will only have access if the memory is configured as data RAM for the CLA. If it is programmed as program RAM, all the accesses from the CPU (including read) and data accesses from the CLA will be blocked, and the violation will be considered a non-master access violation. If the memory is configured as dedicated to the CPU, all accesses from the CLA will be blocked and the violation will be considered a non-master access violation.

### 3.11.1.7 Memory Error Detection, Correction and Error Handling

These devices have memory error detection and correction features to satisfy safety standards requirements. These requirements warrant the addition of detection mechanisms for finite dangerous failures.

In this device, all dedicated RAMs support error correction code (ECC) protection and the shared RAMs have parity protection. The ECC scheme used is Single Error Correction Double Error Detection (SECCDED). The parity scheme used is even parity. ECC/Parity will cover the data bits stored in memory as well as address.

ECC/Parity calculation is done inside the memory controller module and calculated. ECC/Parity is written into the memory along with the data. ECC/Parity is computed for 16-bit data; hence, for each 32-bit of data, there will be three 7-bit ECC codes (or 3-bit parity), two of which are for data and a third one for the address.

#### 3.11.1.7.1 Error Detection and Correction

Error detection is done while reading the data from memory. The error detection is performed for data as well as address. For parity memory, only a single-bit error gets detected, whereas in case of ECC memory, along with a single-bit error, a double-bit error also gets detected. These errors are called correctable and uncorrectable errors. The following are characteristics of these errors:

- Parity errors are always uncorrectable errors
- Single-bit ECC errors are correctable errors
- Double-bit ECC errors are uncorrectable errors
- Address ECC errors are also uncorrectable errors

Correctable errors get corrected by the memory controller module and then correct data is given back as read data to the master. It is also written back into the memory to prevent a double-bit error due to another single-bit error at the same memory address.

#### 3.11.1.7.2 Error Handling

For each correctable error, the count in the correctable error count register will increment by one. When the value in this count register becomes equal to the value configured in the correctable error threshold register, an interrupt is generated to the CPU, if the interrupt is enabled in the correctable interrupt enable register. The user needs to configure the correctable error threshold register based on the system requirements. Also, the address for which the error occurred, gets latched into a register and a flag also gets set in a status register.

If there are uncorrectable errors, an NMI gets generated for the CPU. In this case also, the address for which the error occurred gets latched into a register, and a flag gets set in a status register.

Table 3-12 summarizes different error situations that can arise. These need to be handled appropriately in the software, using the status and interrupt indications provided.

**Table 3-12. Error Handling in Different Scenarios**

Access Type	Error Found In	Error Type	Status Indication	Error Notification
Reads	Data read from memory	Uncorrectable Error (Single-bit error for Parity RAMs OR Double bit Error for ECC RAMs)	Yes - CPU/CPU.DMA/CPU.CLA1 CPU/DMA/CLA Read Error Address Register Data returned to CPU/CPU.DMA/CPU.CLA1 is incorrect	NMI for CPU access NMI for CPU.DMA access NMI to CPU for CPU.CLA1 access
Reads	Data read from memory	Single-bit error for ECC RAMs	Yes - CPU/CPU.DMA CPU/DMA Read Error Address Register Increment single error counter	Interrupt when error counter reaches the user programmable threshold for single errors
Reads	Address	Address error	Yes - CPU/CPU.DMA/CPU.CLA1 CPU/DMA/CLA Read Address Error Register Data returned to CPU/CPU.DMA/CPU.CLA1 is incorrect	NMI to CPU for CPU access NMI to CPU for CPU.DMA access NMI to CPU for CPU.CLA1 access

---

**NOTE:** In the case of an uncorrectable error during fetch on the CPU, there is the possibility of getting an ITRAP before an NMI exception, since garbage instructions enter into the CPU pipeline before the NMI gets generated.

During debug accesses, correctable as well as uncorrectable errors are masked.

---

### 3.11.1.8 Application Test Hooks for Error Detection and Correction

Since error detection and correction logic is part of safety critical logic, safety applications may need to ensure that the logic is always working fine (during run time also). To enable this, a test mode is provided, in which a user can modify the data bits (without modifying the ECC/Parity bits) or ECC/Parity bits directly. Using this feature, an ECC/Parity error could be injected into data.

---

**NOTE:** The memory map for ECC/Parity bits and data bits are the same. The user must choose a different test mode to access ECC/Parity bits. In test mode, all access to memories (data as well as ECC/Parity) should be done as 32-bit access only.

---

The following table shows the bit mapping for the ECC/Parity bits when they are read in RAMTEST mode using their respective addresses.

**Table 3-13. Mapping of ECC Bits in Read Data from ECC/Parity Address Map**

Data Bits Location in Read Data	Content (ECC Memory)
6:0	ECC Code for lower 16 bits of data
7	Not Used
14:8	ECC Code for upper 16 bits of data
15	Not Used
22:16	ECC Code for address
31:23	Not Used

**Table 3-14. Mapping of Parity Bits in Read Data from ECC/Parity Address Map**

Data Bits Location in Read Data	Content (Parity Memory)
0	Parity for lower 16 bits of data
7:1	Not Used
8	Parity for upper 16 bits of data
15:9	Not Used
16	Parity for address
31:17	Not Used

### 3.11.1.9 RAM Initialization

To ensure that read/fetch from uninitialized RAM locations do not cause ECC or parity errors, the RAM\_INIT feature is provided for each memory block. Using this feature, any RAM block can be initialized with 0x0 data and respective ECC/Parity bits accordingly. This can be initiated by setting the INIT bit to '1' for the specific RAM block in INIT registers. To check the status of RAM initialization, SW should poll for the INITDONE bit for that RAM block in the INITDONE register to be set. Unless this bit gets set, no access should be made to that RAM memory block.

---

**NOTE:** None of the masters should access the memory while initialization is taking place. If memory is accessed before RAMINITDONE is set, the memory read/write as well as initialization will not happen correctly.

---



## 3.12 Flash and OTP Memory

Flash is an electrically erasable/programmable nonvolatile memory that can be programmed and erased many times to ease code development. Flash memory can be used primarily as a program memory for the core, and secondarily as static data memory.

This section describes the proper sequence to configure the wait states and operating mode of flash. It also includes information on flash and OTP power modes, how to improve flash performance by enabling the flash prefetch/cache mode, and the SECDED safety feature.

### 3.12.1 Features

Features of flash memory include:

- Up to two flash banks (Bank0 and Bank1) (refer to the device data manual for the number and size of the flash banks)
- One FMC controlling up to two flash banks
- 128 bits (bank width) can be programmed at a time along with ECC
- Multiple sectors providing the option of leaving some sectors programmed and only erasing specific sectors
- User-programmable OTP locations (in user-configurable DCSM OTP, also called USER OTP) for configuring security, OTP boot-mode and boot-mode select pins (if the user is unable to use the factory-default boot-mode select pins)
- Flash pump shared by the two banks
- Enhanced performance using the code-prefetch mechanism and data cache in FMC
- Configurable wait states to give the best performance for a given execution speed
- Safety Features
  - SECDED-single error correction and double error detection is supported in the FMC
  - Address bits are included in ECC
  - Test mode to check the health of ECC logic
- Supports low-power modes for flash bank and pump for power savings
- Built-in power mode control logic
- Integrated flash program/erase state machine (FSM) in the FMC
  - Simple flash API algorithms
  - Fast erase and program times (refer to the device data manual for details)
- Dual Code Security Module (DCSM) to prevent access to the flash by unauthorized persons (refer to [DCSM](#) for details)

### 3.12.2 Flash Tools

Texas Instruments provides the following tools for flash:

- Code Composer Studio (CCS) - the development environment with integrated flash plugin
- F021 Flash API Library - a set of software peripheral functions to erase/program flash  
Please refer to the [TMS320F28004x Flash API Reference Guide](#) for more information.
- UniFlash - standalone tool to erase/program/verify the flash content through JTAG. No CCS is required.
- Linker ECC generation - to generate the Flash ECC from the Flash data and to introduce errors in Flash/ECC space. Please refer to the ([TMS320C28x Assembly Language Tools User's Guide](#)) for more information.
- CCS On-Chip Flash Plugin and UniFlash tools are developed for a fully Flash-embedded application, which should be a standalone program that has all the initialized sections linked to flash. Any code that needs to be executed from RAM should be copied from Flash at run time.
- Users must check and install available updates for CCS On-Chip Flash Plugin and UniFlash tools.

### 3.12.3 Default Flash Configuration

The following are flash module configuration settings at power-up:

- Flash banks are in sleep power mode (BNKPWR bit field in the FBFALLBAC register)
- Shared pump is in sleep power mode (PMPPWR bit field in the FPAC1 register)
- ECC is enabled
- Wait-states are set to the maximum (0xF)
- Code-prefetch mechanism and data cache are disabled in the FMC
- Bank and pump active grace periods are set to 0x0 (refer to the BAGP field in the FBAC register and PAGP bit field in the FPAC2 register)

Note that boot ROM changes the BNKPWR and PMPPWR bit fields to active mode.

User application software must initialize wait-states using the FRDCNTL register, and configure cache/prefetch features using the FRD\_INTF\_CTRL register, to achieve optimum system performance. Software that configures flash settings like wait-states, cache/prefetch features, and so on, must be executed only from RAM memory, **not** from flash memory.

---

**NOTE:** Before initializing wait-states, turn off the pre-fetch and data caching in the FRD\_INTF\_CTRL register.

---

### 3.12.4 Flash Bank, OTP and Pump

There are two flash banks, Bank0 and Bank1 (refer to the device data manual for the number of banks available). Also, there is a one-time programmable (OTP) memory called USER OTP, which the user can program only once and cannot erase. Flash and OTP are uniformly mapped in both program and data memory space.

There is also a TI-OTP which contains manufacturing information like settings used by the flash state machine for erase and program operations, and so on. Users may read TI-OTP but it cannot be programmed or erased. For memory map and size information of the banks, TI-OTP, USER OTP, and corresponding ECC locations, please refer to the device data manual.

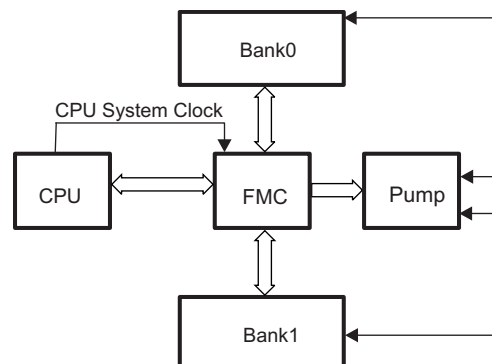
Bank0 and Bank1 share a common flash pump; therefore, only one bank can be programmed or erased at a time. Execution or reads from one bank, while erase or program is in progress on the other bank, is supported.

Figure 3-19 depicts the user-programmable OTP locations in USER-OTP . For more information on the functionality of these fields, please refer to [DCSM](#) and the *ROM Code and Peripheral Booting* chapter.

### 3.12.5 Flash Module Controller (FMC)

A single FMC controls both Bank0 and Bank1. The CPU interfaces with the FMC, which in turn interfaces with Bank0 and Bank1 and the shared pump, to perform erase or program operations, to read data, and execute code from these flash banks.

Figure 3-15. FMC Interface with Core, Bank and Pump



There is a state machine in FMC which generates the erase/program sequences in hardware. This simplifies the Flash API software which configures control registers in the FMC to perform flash erase and program operations. Please refer to the [TMS320F28004x Flash API Reference Guide](#) for more information for details on Flash API.

[Section 3.12.6](#) through [Section 3.12.10](#) describe FMC in detail.

### 3.12.6 Flash and OTP and Wakeup Power-Down Modes

The flash banks and pump consume a significant amount of power when active. The flash module provides a mechanism to power-down flash banks and pump. Special timers automatically sequence the power-up and power-down of Bank0 and Bank1 independently of each other. The shared charge pump module has its own independent power-up and power-down timers as well.

#### 3.12.6.1 Flash/OTP and Pump Power Modes and Wakeup

The flash banks and OTP operate in three power modes: Sleep (lowest power), Standby, and Active (highest power)

- **Sleep State**  
This is the state after a device reset. In this state, a CPU data read or opcode fetch will automatically initiate a change in power mode to the standby state and then to the active state. During this transition time to the active state, the CPU will automatically be stalled.
- **Standby State**  
This state uses more power than the sleep state, but takes a shorter time to transition to the active or read state. In this state, a CPU data read or opcode fetch will automatically initiate a change in power mode to the active state. During this transition time to the active state, the CPU will automatically be stalled. Once the flash/OTP has reached the active state, the CPU access will complete as normal.
- **Active State**  
In this state, the bank and pump are in active power mode state (highest power)

The charge pump operates in two power modes:

- Sleep (lowest power)
- Active (highest power)

Any access to any flash bank/OTP causes the charge pump to go into active mode, if it is in sleep mode. Also, any erase or program command causes the charge pump and bank to become active. If any bank is active or in standby mode, the charge pump will be in active mode, independent of the charge pump power mode control configuration (refer to the PMPPWR bit field in the FPAC1 register). While the pump is in sleep state, a charge pump sleep down counter holds a user-configurable value (PSLEEP bit field in the FPAC1 register) and when the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP prescaled SYSCLK clock cycles (prescaled clock is SYSCLK/2) before putting the charge pump into active power mode. Note that the configured PPSLEEP value should yield at least a delay of 20  $\mu$ s for the pump to go to active mode. Refer to the register descriptions, [Registers](#), for detailed information.

Below are the number of cycles it will take for the bank and pump to wake up from low power modes.

1. Pump sleep to active = PSLEEP \* (SYSCLK/2) cycles
2. Bank sleep to standby = 254 Flash clock cycles
3. Bank standby to active = 55 Flash clock cycles

Where in Flash clock = SYSCLK/(RWAIT+1)

#### 3.12.6.2 Active Grace Period

The active grace period (AGP) can be used to optimize the flash module power consumption versus access time. Faster access times are associated with higher-power modes of operation. At one extreme, the power control logic could attempt to reduce power consumption by putting the bank and charge pump into a low-power mode immediately at the end of every flash access. However, if accesses are only a few cycles apart, this can actually increase power consumption versus leaving the flash powered, because the bank and charge pump consume more power during flash startup and access.

The active grace periods (supported for Bank0 and Bank1 together, in addition to the charge pump module) allow the banks and/or charge pump to be maintained in active mode for a specified period following an access. This is done in anticipation of another read within the AGP time, to allow the subsequent read to have a faster access and spend less time dissipating power, than if the bank went into one of the low power modes immediately. If the next access does not occur within the AGP time, the power control logic can automatically put the bank and/or charge pump into a low-power mode to reduce power consumption during long periods of inactivity.

The AGP value is programmed by a set of programmable counters (FBAC and FPAC2) which keep the flash bank or charge pump in active mode until the counter expires, at which time the bank or charge pump reverts to its fallback power mode as defined in the FBFALLBACK and FPAC1 (refer to PMPPWR bit-field) registers. The application software can configure the fallback power mode to reduce power consumption, or configure it to be active mode to keep the bank active regardless of counter settings (default is SLEEP). The charge pump AGP counter remains in its initialized state when the bank is active, including the AGP counter of the bank. The charge pump AGP counter begins counting when the bank has become inactive.

The application software can check the current power mode of flash bank and charge pump by reading the FBPRDY register. Refer to the register descriptions, [Registers](#), for detailed information.

### 3.12.7 Flash and OTP Performance

Once the flash bank and pump are in the active power state, a read or fetch access can be classified as a flash access (access to an address location in flash) or an OTP access (access to an address location in OTP). Once the CPU throws an access to a flash memory address, data is returned after RWAIT+1 number of SYSCLK cycles. For a USER-OTP access, data is returned after 11 SYSCLK cycles.

RWAIT defines the number of random access wait-states and is configurable using the RWAIT bit-field in the FRDCNTL register. At reset, the RWAIT bit-field defaults to a worst-case wait-state count (15), and therefore needs to be initialized for the appropriate number of wait states to improve performance, based on the CPU clock rate and the access time of the flash.

For a given system clock frequency, RWAIT has to be configured using below formula:

$$\text{RWAIT} = \text{ceiling}[(\text{SYSCLK}/\text{FCLK})-1]$$

where SYSCLK is the system operating frequency

FCLK is flash clock frequency. FCLK should be  $\leq \text{FCLK}_{\text{max}}$ , allowed maximum flash clock frequency at RWAIT=0.

If RWAIT results in a fractional value when calculated using the above formula, RWAIT has to be rounded up to the nearest integer. Please see the device data manual for the RWAIT configuration details.

### 3.12.8 Flash Access Interface

This section provides details about the modes to access flash/OTP and the configuration registers which control the read interface. In addition to a standard read mode, the FMC has a built-in prefetch and cache mechanism to allow increased clock speeds and CPU throughput wherever applicable.

#### 3.12.8.1 Standard Access Mode

Standard access mode is defined as the access mode in effect when the code prefetch-mechanism and data cache are disabled. It is also the default mode after reset. During this mode, each access to flash is decoded by the flash wrapper to read/fetch the data/code from the addressed location and the data/code is returned after the RWAIT+1 number of cycles.

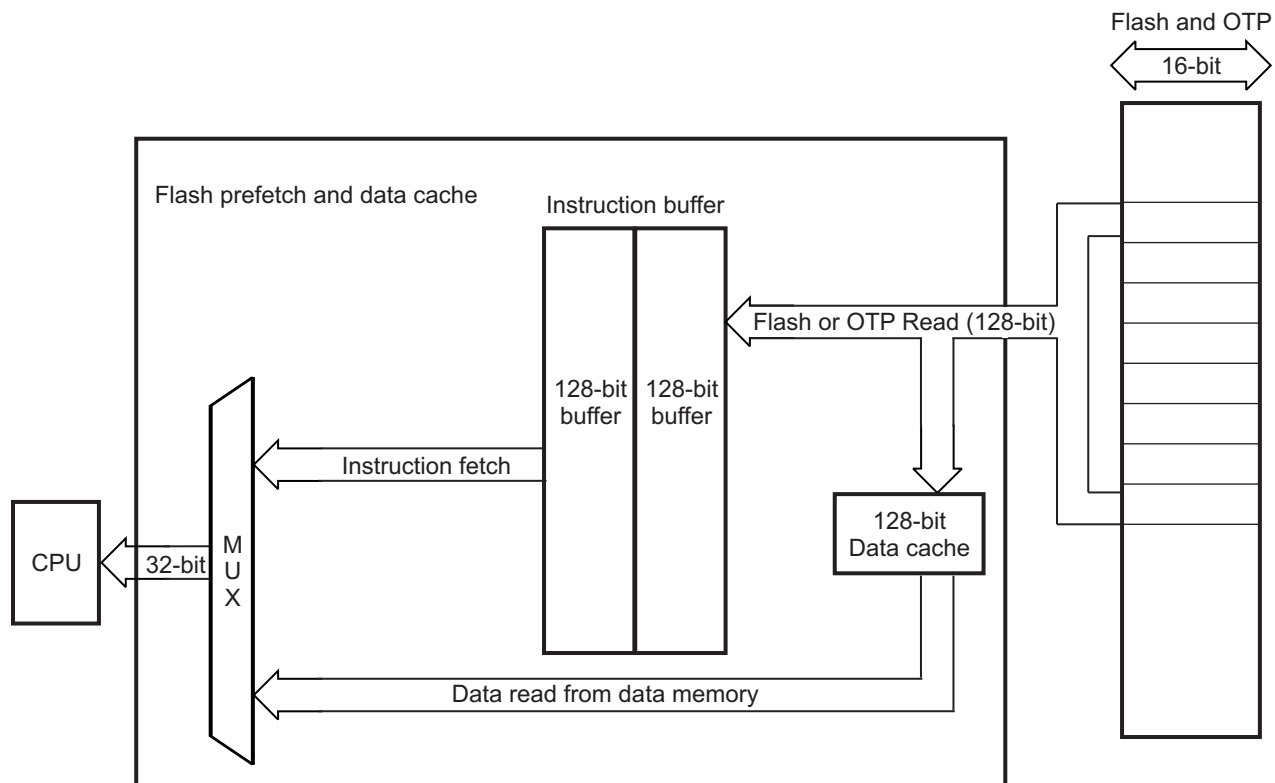
The prefetch buffer associated with the prefetch mechanism and data cache is bypassed in standard access mode; therefore, every access to the flash/OTP is used by the CPU immediately, and every access creates a unique flash bank access.

Standard access mode is the recommended mode for lower system frequency operation in which RWAIT can be set to zero to provide single-cycle access operation. The FMC can operate at higher frequencies using standard access mode at the expense of adding wait states. At higher system frequencies, it is recommended to enable cache and prefetch mechanisms to improve performance. Refer to the device-specific data manual to determine the maximum flash frequency allowed in standard access mode (that is, maximum flash clock frequency with RWAIT=0, FCLK<sub>max</sub>).

### 3.12.8.2 Prefetch Mode

Flash memory is typically used to store application code. During code execution, instructions are fetched from sequential memory addresses, except when a discontinuity occurs. Usually the portion of the code that resides in sequential addresses makes up the majority of the application code and is referred to as linear code. To improve the performance of linear code execution, a flash prefetch-mechanism has been implemented in the FMC. Figure 3-16 illustrates how this mode functions.

**Figure 3-16. Flash Prefetch Mode**



This prefetch mechanism does a look-ahead prefetch on linear address increments starting from the address of the last instruction fetch. The flash prefetch mechanism is disabled by default. Setting the PREFETCH\_EN bit in the FRD\_INTF\_CTRL register enables this prefetch mode.

An instruction fetch from the flash or OTP reads out 128 bits per access. The starting address of the access from flash is automatically aligned to a 128-bit boundary, such that the instruction location is within the 128 bits to be fetched. With the flash prefetch mode enabled, the 128 bits read from the instruction buffer are stored in a 128-bit wide by 2-level deep instruction prefetch buffer. The contents of this prefetch buffer are then sent to the CPU for processing as required.

Up to four 32-bit or eight 16-bit instructions can reside within a single 128-bit access. The majority of C28x instructions are 16 bits, so for every 128-bit instruction fetch from the flash bank, it is likely that there are up to eight instructions in the prefetch buffer ready to process through the CPU. During the time it takes to process these instructions, the flash prefetch mechanism automatically initiates another access to the flash bank to prefetch the next 128 bits. In this manner, the flash prefetch mechanism works in the background to keep the instruction prefetch buffers as full as possible. Using this technique, the overall efficiency of sequential code execution from flash or OTP is improved significantly.

---

**NOTE:** If the prefetch mechanism is enabled, then the last two rows (16 16-bit words i.e. 256 bits) of the bank which does not have valid address beyond its boundary should not be used, because the prefetch logic which does a look-ahead prefetch, will try to fetch from outside the bank and would result in an ECC error.

---

The flash prefetch is aborted only on a PC discontinuity caused by executing an instruction such as a branch, BANZ, call, or loop. When this occurs, the prefetch mechanism is aborted and the contents of the prefetch buffer are flushed. There are two possible scenarios when this occurs:

1. If the destination address is within the flash or OTP, the prefetch aborts and then resumes at the destination address.
2. If the destination address is outside of the flash and OTP, the prefetch is aborted and begins again only when a branch is made back into the flash or OTP. The flash prefetch mechanism only applies to instruction fetches from program space. Data reads from data memory and from program memory do not utilize the prefetch buffer capability and thus bypass the prefetch buffer. For example, instructions such as MAC, DMAC, and PREAD read a data value from program memory. When this read happens, the prefetch buffer is bypassed but the buffer is not flushed. If an instruction prefetch is already in progress when a data read operation is initiated, then the data read will be stalled until the prefetch completes.

Note that the prefetch mechanism gets bypassed when RWAIT is configured as zero.

### 3.12.8.3 Data Cache

Along with the prefetch mechanism, a data cache of 128 bits wide is also implemented to improve data-space read and program-space read performance. This data cache will not be filled by the prefetch mechanism. When any kind of data-space read or program-space read is made by the CPU from an address in the bank, and if the data corresponding to the requested address is not in the data cache, then 128 bits of data will be read from the bank and loaded in the data cache. This data is eventually sent to the CPU for processing. The starting address of the access from flash is automatically aligned to a 128-bit boundary such that the requested address location is within the 128 bits to be read from the bank. By default, this data cache is disabled and can be enabled by setting DATA\_CACHE\_EN bit in the FRD\_INTF\_CTRL register. Note that the data cache gets bypassed when RWAIT is configured as zero.

Some other points to keep in mind when working with flash/ OTP:

- Reads of the USER OTP locations are hardwired for 10 wait states. The RWAIT bits have no effect on these locations.
- CPU writes to the flash or OTP memory map areas are ignored. They complete in a single cycle.
- If a security zone is in the locked state and the respective password lock bits are not all 1s, then,
  - Data reads to Zx-CSMPSWD will return 0
  - Program space reads to Zx-CSMPSWD will return 0
  - Program fetches to Zx-CSMPSWD will return 0
- When the Dual Code Security Module (DCSM) is secured, reads to the flash/OTP memory map area from outside the secure zone take the same number of cycles as a normal access. However, the read operation returns a zero.
- The arbitration scheme in FMC prioritizes CPU accesses in the fixed priority order of data read (highest priority), program space read and program fetches/program prefetches (lowest priority).
- When FSM interface is active for erase/program operations, data in the prefetch buffers and data cache in FMC will be flushed.

When the data cache is enabled, the debugger memory window open to Flash/OTP space will invoke data caching. Therefore, the debugger memory window should not be left open for Flash/OTP space when benchmarking the code for performance.



### 3.12.9 Erase/Program Flash

Flash memory may be programmed either by using the CCS Flash plugin or by using Uniflash. If these methods are not feasible in an application, the API may be used. The Flash memory should be programmed, erased, and verified only by using the F021 Flash API library. These functions are written, compiled and validated by Texas Instruments. The flash module contains a flash state machine (FSM) to perform program and erase operations. This section only provides a high-level description for these operations; therefore, please refer to the [TMS320F28004x Flash API Reference Guide](#) for more information.

Note that Flash API execution is interruptible. However, there should not be any read/fetch access from the Flash bank on which an erase/program operation is in progress. In single-bank devices, Flash API must be executed from RAM. In dual-bank devices Flash API can execute from one bank to perform erase/program operations on another bank. If prefetch is enabled, note that the last 128 bits of Bank0 should not be accessed when erasing/programming Bank1 since accessing them will cause a prefetch access to Bank1.

A typical flow to program flash is:  
Erase → Program → Verify

Always refer to the device-specific support folder in C2000Ware for the latest Flash API library.

#### 3.12.9.1 Erase

When the target flash is erased, it reads as all 1's. This state is called 'blank.' The erase function must be executed before programming. The user should NOT skip erase on sectors that read as 'blank' because these sectors may require additional erasing due to marginally erased bits columns. The FSM provides an "Erase Sector" command to erase the target sector. The erase function erases the data and the ECC together. This command is implemented by the following Flash API function:

```
Fapi_issueAsyncCommandWithAddress();
```

The Flash API provides the following function to determine if the flash bank is 'blank':

```
Fapi_doBlankCheck();
```

#### 3.12.9.2 Program

The FSM provides a command to program the USER OTP and Flash. This command is also used to program ECC check bits.

This command is implemented by the following Flash API function:

```
Fapi_issueProgrammingCommand();
```

The Program function provides the options to program data without ECC, data with user-provided ECC data with ECC calculated by API software, and to program ECC only.

---

**NOTE:** The main array flash programming must be aligned to 64-bit address boundaries and each 64-bit word may only be programmed once per write/erase cycle.

The DCSM OTP programming must be aligned to 128 bit address boundaries and each 128 bit word may only be programmed once. The exceptions are:

- The DCSM Zx-LINKPOINTER1 and Zx-LINKPOINTER2 values in the DCSM OTP should be programmed together, and may be programmed 1 bit at a time as required by the DCSM operation.
  - The DCSM Zx-LINKPOINTER3 values in the DCSM OTP may be programmed 1 bit at a time on a 64-bit boundary to separate it from Zx-PSWDLOCK, which must only be programmed once.
- 

#### 3.12.9.3 Verify

After programming, the user can verify the programmed contents using API function `Fapi_doVerify()`. This function verifies the flash contents against supplied data.



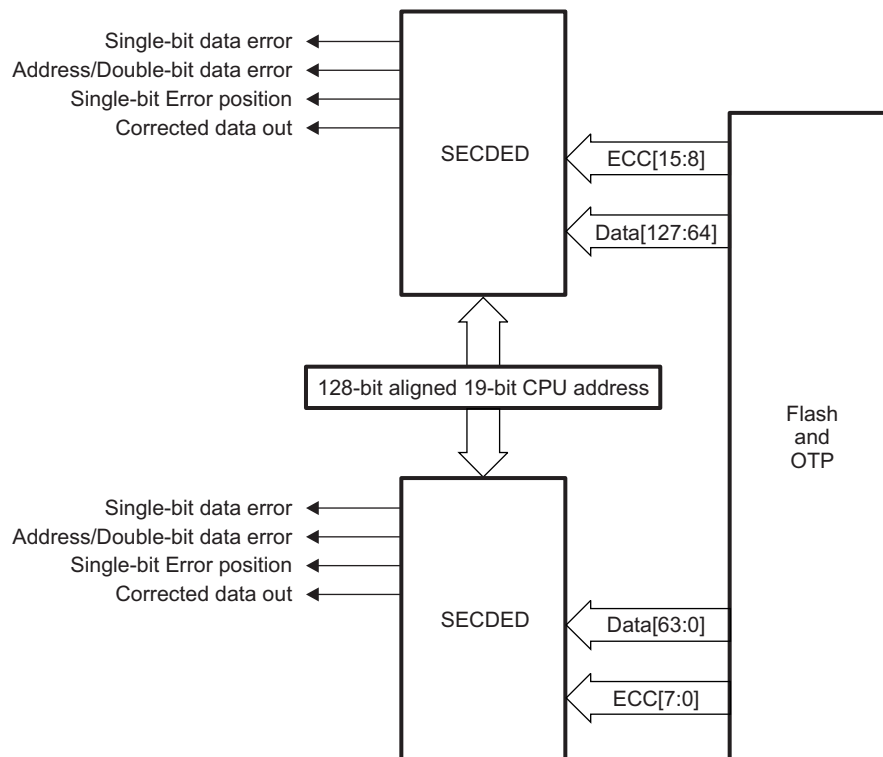
Application software typically perform a CRC check of the Flash memory contents during power-up and at regular intervals during runtime (as needed). Apart from this, ECC logic, when enabled (enabled by default) catches single-bit errors, double-bit errors, and address errors whenever the CPU reads or fetches from a Flash address.

### 3.12.10 Error Correction Code (ECC) Protection

FMC contains an embedded single error correction and double error detection (SECDED) module. SECDED, when enabled, provides the capability to screen out memory faults. SECDED can detect and correct single-bit data errors and detect address errors/double-bit data errors. For every 64 bits of flash/OTP data (aligned on a 64-bit memory boundary) that is programmed, eight ECC check bits have to be calculated and programmed in ECC memory space. Refer to the device data manual for the Flash/OTP ECC memory map. ECC memory space in Flash should not be used to program code or data; it is used only for error correction and detection. SECDED works with a total of eight user-calculated error correction code (ECC) check bits associated with each 64-bit wide data word and its corresponding 128-bit memory-aligned address. Users must program ECC check bits along with flash. TI recommends using the AutoEccGeneration option available in Plugin/API to program ECC. Users can use the F021 Flash API to calculate and program ECC data along with flash data. Flash API uses hardware ECC logic in the device to generate the ECC data for the given Flash data. The Flash Plugin, the Flash programming tool integrated with Code Composer Studio, uses Flash API to generate and program ECC data. Alternatively users can use the linker-supported ECC generation option to generate and place ECC in separate sections through the linker command file. Please refer to TMS320C28x Assembly Language Tools User's Guide for more information on using the linker-supported ECC generation method.

Figure 3-17 illustrates the ECC logic inputs and outputs.

Figure 3-17. ECC Logic Inputs and Outputs



During an instruction fetch or a data read operation, the 19 most significant address bits (three least significant bits of address are not considered), together with the 64-bit data/8-bit ECC read-out of flash banks/ECC memory map area, pass through the SECDED logic and the eight checkbits are produced in FMC. These eight calculated ECC check bits are then XORed with the stored check bits (user programmed check bits) associated with the address and the read data. The 8-bit output is decoded inside the SECDED module to determine one of three conditions:

- No error occurred
- A correctable error (single bit data error) occurred
- An uncorrectable error (double-bit data error or address error) occurred

If the SECDED logic finds a single-bit error in the address field, then it is considered to be an uncorrectable error.

---

**NOTE:** Since ECC is calculated for an entire 64-bit data, a non 64-bit read such as a byte read or a half-word read will still force the entire 64-bit data to be read and calculated, but only the byte or half-word will be actually used by the CPU.

---

This ECC (SECDED) feature is enabled at reset. The ECC\_ENABLE register can be used to configure (enable/disable) the ECC feature. The ECC for the application code must be programmed. There are two SECDED modules in each FMC. Out of the 128-bit data (aligned on a 128-bit memory boundary) read from the bank/OTP address, the lower 64-bits of data and corresponding 8 ECC bits (read from user programmable ECC memory area) are fed as inputs to one SECDED module along with 128-bit aligned 19-bit address from where data has been read. The upper 64-bits of data and corresponding 8 ECC bits are fed as inputs to another SECDED module in parallel, along with 128-bit aligned 19-bit address. Each of the SECDED modules evaluate their inputs and determine if there is any single-bit data error or double-bit data error/address error.

ECC logic will be bypassed when the 64 data bits and the associated ECC bits fetched from the bank are either all ones or zeros.

### 3.12.10.1 Single-Bit Data Error

This section provides information for both single-bit data errors and single-bit ECC check bit errors. If there is a single bit flip (0 to 1 or 1 to 0) in flash data or in ECC data, then it is considered as a single-bit data error. The SECDED module detects and corrects single-bit errors, if any, in the 64-bit flash data or eight ECC check bits read from the flash/ECC memory map before the read data is provided to the CPU.

When SECDED finds and corrects single bit data errors, the following information is logged in the ECC registers if the ECC feature is enabled:

- Address where the error occurred – if the single-bit error occurs in the lower 64-bits of a 128-bit memory-aligned data, the lower 64-bit memory-aligned address will be captured in the SINGLE\_ERR\_ADDR\_LOW register. If the single-bit error occurs in the upper 64-bits of a 128-bit memory-aligned data, the upper 64-bit memory-aligned address will be captured in the SINGLE\_ERR\_ADDR\_HIGH register.
- Whether the error occurred in data bits or ECC bits – the ERR\_TYPE\_L and ERR\_TYPE\_H bit fields in the ERR\_POS register indicate whether the error occurred in data bits or ECC bits of the lower 64-bits, or the upper 64-bits respectively, of a 128-bit memory-aligned data.
- Bit position at which error occurred – the ERR\_POS\_L and ERR\_POS\_H bit fields in the ERR\_POS register indicate the bit position of the error in the lower 64-bits/lower 8-bit ECC, or the upper 64-bits/upper 8-bit ECC respectively, of a 128-bit memory-aligned data.
- Whether the corrected value is 0 (FAIL\_0\_L, FAIL\_0\_H flags in ERR\_STATUS register)
- Whether the corrected value is 1 (FAIL\_1\_L, FAIL\_1\_H flags in ERR\_STATUS register)
- A single bit error counter that increments on every single bit error occurrence (ERR\_CNT register) until a user-configurable threshold (see ERR\_THRESHOLD) is met
- A flag that gets set when one or more single-bit errors occurs after ERR\_CNT equals ERR\_THRESHOLD (SINGLE\_ERR\_INT\_FLG flag in the ERR\_INTFLG register)

When the ERR\_CNT value equals THRESHOLD+1 value and a single bit error occurs, the SINGLE\_ERR\_INT flag is set, and an interrupt (FLASH\_CORRECTABLE\_ERR on C28x PIE has to be enabled for interrupt, if needed) is fired. The SINGLE\_ERR interrupt will not be fired again until the SINGLE\_ERR\_INTFLG is cleared. If the single error interrupt flag is not cleared using the corresponding error interrupt clear bit in the ERR\_INTCLR register, the error interrupt will not come again, as this is an edge-based interrupt.

When multiple single-bit errors get caught by ECC logic, the Flash ECC error registers will hold the information related to the latest ECC error. Although ECC is calculated on a 64-bit basis, a read of any address location within a 128-bit aligned flash memory will cause the single-bit error flag to get set when there is a single-bit error in both or in either one of the lower 64 and upper 64 bits (or corresponding ECC check bits) of that 128-bit data.

### 3.12.10.2 Uncorrectable Error

Uncorrectable errors include address errors and double-bit errors in data/ECC. When SECEDED finds uncorrectable errors, the following information is logged in ECC registers if the ECC feature is enabled:

- Address where the error occurred – if the uncorrectable error occurs in the lower 64-bits of a 128-bit memory-aligned data, the lower 64-bit memory-aligned address will be captured in the UNC\_ERR\_ADDR\_LOW register. If the uncorrectable error occurs in the upper 64-bits of a 128-bit memory-aligned data, the upper 64-bit memory-aligned address will be captured in the UNC\_ERR\_ADDR\_HIGH register.
- A flag is set indicating that an uncorrectable error occurred – the UNC\_ERR\_L and UNC\_ERR\_H flags in the ERR\_STATUS register indicate the uncorrectable error occurrence in the lower 64-bits/lower 8-bit ECC, or the upper 64-bits/upper 8-bit ECC, respectively, of a 128-bit memory-aligned data.
- A flag is set indicating that an uncorrectable error interrupt is fired (UNC\_ERR\_INTFLG in ERR\_INTFLG register)

When an uncorrectable error occurs, the UNC\_ERR\_INTFLG bit is set and an uncorrectable error interrupt is fired. This uncorrectable error interrupt generates an NMI, if enabled. If an uncorrectable error interrupt flag is not cleared using the corresponding error interrupt clear bit in the ERR\_INTCLR register, an error interrupt will not come again, as this is an edge based interrupt.

Although ECC is calculated on 64-bit basis, a read of any address location within a 128-bit aligned Flash memory will cause the uncorrectable error flag to get set when there is a uncorrectable error in both or in either one of the lower 64 and upper 64 bits (or corresponding ECC check bits) of that 128-bit data. NMI will occur on the CPU for a read of any address location within a 128-bit aligned Flash memory, when there is an uncorrectable error in both or in either one of the lower 64 and upper 64 bits (or corresponding ECC check bits) of that 128-bit data.

### 3.12.10.3 SECEDED Logic Correctness Check

Since error detection and correction logic are part of safety-critical logic, safety applications may need to ensure that the SECEDED logic is always working properly. For these safety concerns, in order to ensure the correctness of the SECEDED logic, an ECC test mode is provided to test the correctness of ECC logic periodically. In ECC test mode, Data/ECC and address inputs to the ECC logic are controlled by the ECC test mode registers FDATAH\_TEST, FDATAL\_TEST, FECC\_TEST, and FADDR\_TEST, respectively. Using this test mode, users can introduce single-bit errors, double-bit errors, or address errors and check whether or not SECEDED logic is catching those errors. Users can also check if SECEDED logic is reporting any false errors when no errors are introduced.

This ECC test mode can be enabled by setting the ECC\_TEST\_EN bit in the FECC\_CTRL register. When ECC test mode is enabled, the CPU cannot read the data from flash and instead the CPU gets data from the ECC test mode registers (FDATAH\_TEST/FDATAL\_TEST). This is because ECC test mode registers (FDATAH\_TEST, FDATAL\_TEST, FECC\_TEST) are multiplexed with data from the flash. Hence, the CPU should not read/fetch from Flash when ECC test mode is enabled. For this reason, ECC test mode code should be executed from RAM and not from flash.

Only one of the SECEDED modules (out of the two SECEDED modules that work on lower 64 bits and upper 64 bits of a read 128-bit data) at a time can be tested. The ECC\_SELECT bit in the FECC\_CTRL register can be configured by users to select one of the SECEDED modules for test.

To test the ECC logic using ECC test mode, users can follow the steps below:

1. Obtain the ECC for a given Flash address (128-bit aligned) and 64-bit data by using Auto ECC generation option provided in Flash API or by using the linker ECC generation method.
2. Develop an application to test ECC logic using the above data. In this application
  - Write the 128-bit aligned 19-bit Flash address in FADDR\_TEST

- Write 64-bit data in FDATAH\_TEST (upper 32 bits) and FDATAL\_TEST (lower 32 bits) registers
- Write the corresponding 8-bit ECC in the FECC\_TEST register
- In any of the above three steps, users can insert errors (single-bit data error or double-bit data error or address error or single-bit ECC error or double-bit ECC error) so that they can check whether or not ECC logic is able to catch the errors
- Select the ECC logic block (lower 64-bits or upper 64-bits) which needs to be tested using the ECC\_SELECT bit in the FECC\_CTRL register
- Enable ECC test mode using the ECC\_TEST\_EN bit in FECC\_CTRL register
- Write a value of 1 in the DO\_ECC\_CALC bit in FECC\_CTRL register to enable ECC test logic for a single cycle to evaluate the address, data, ECC in FADDR\_TEST, FDATAx\_TEST and FECC\_TEST registers for ECC errors

Once the above ECC test mode registers are written by the user:

- The FECC\_OUTH register holds the data output bits 63:32 from the SECDED block under test
- The FECC\_OUTL register holds the data output bits 31:0 from the SECDED block under test
- The FECC\_STATUS register holds the status of single-bit error occurrence, uncorrectable error occurrence, and error position of single-bit error in data/check bits

#### 3.12.10.4 Reading ECC Memory From a Higher Address Space

In these devices, ECC memory for Flash and OTP is allocated at a higher address space (address width more than 22 bits). C2000 Codegen tools (6.2 and onwards) are updated to include the below intrinsics to read ECC space.

For 16-bit read:

```
unsigned int variable = __addr32_read_uint16(unsigned long address);
```

For 32-bit read:

```
unsigned long variable = __addr32_read_uint32(unsigned long address);
```

#### 3.12.11 Reserved Locations Within Flash and OTP

When allocating code and data to flash and OTP memory, keep the following reserved locations in mind:

- The entire OTP has reserved user-configurable locations for security and boot process. For more details on the functionality of these fields, please refer to [Section 3.13](#), and the *ROM Code and Peripheral Booting* chapter.
- Refer to the *ROM Code and Peripheral Booting* chapter for reserved locations in flash for real-time operating system usage and a boot-to-flash entry point. A boot-to-flash entry point is reserved for an entry-into-flash branch instruction. When the boot-to-flash boot option is used, the boot ROM will jump to this address in flash. If the user programs a branch instruction here, that will then redirect code execution to the entry point of the application.

#### 3.12.12 Procedure to Change the Flash Control Registers

During flash configuration, no accesses to the flash or OTP can be in progress. This includes instructions still in the CPU pipeline, data reads, and instruction prefetch operations. To be sure that no access takes place during the configuration change, you should follow the procedure shown below for any code that modifies the flash control registers.

1. Start executing application code from RAM/Flash/OTP.
2. Branch to or call the flash configuration code (that writes to flash control registers) in RAM. This is required to properly flush the CPU pipeline before the configuration change. The function that changes the flash configuration cannot execute from the Flash or OTP. It must reside in RAM.
3. Execute the flash configuration code (should be located in RAM) that writes to flash control registers like FRDCNTL, FRD\_INTF\_CTRL, and so on.
4. At the end of the flash configuration code execution, wait eight cycles to let the write instructions propagate through the CPU pipeline. This must be done before the return-from-function call is made.
5. Return to the calling function which might reside in RAM or Flash/OTP and continue execution.

### 3.13 Dual Code Security Module (DCSM)

The dual code security module (DCSM) is a security feature incorporated in this device. It prevents access and visibility to on-chip secure memories (and other secure resources) to unauthorized persons. It also prevents duplication and reverse engineering of proprietary code. The term “secure” implies that access to on-chip secure memories and resources are blocked. The term “unsecure” implies that access is allowed (the contents of the memory could be read by any means); for example, through a debugging tool such as Code Composer Studio™.

The CSM has dual-zone security; zone1 and zone2.

#### 3.13.1 Functional Description

The security module restricts the CPU access to on-chip secure memory and resources without interrupting or stalling CPU execution. When a read occurs to a secure memory location, the read returns a zero value and CPU execution continues with the next instruction. This, in effect, blocks read and write access to secure memories through the JTAG port or external peripherals.

The code security mechanism offers protection for two zones, Zone 1 (Z1) and Zone 2 (Z2). The security mechanism for both the zones is identical. Each zone has its own dedicated secure resource and allocated secure resource. The following are different secure resources available on this device:

- **OTP:** Each zone has its own dedicated secure OTP (USER OTP). This contains the security configurations for the individual zone. If a zone is secure, its USER OTP content (including CSM passwords) can be read (execution not allowed) only if the zone is unlocked using the password match flow (PMF). This device has two Flash banks (Bank0 and Bank1) and each bank has its own USER OTP. Both banks' USER OTP are secure and partitioned between Zone1 and Zone2.
- **RAM:** All LSx RAMs can be secure RAM on this device. These RAMs can be allocated to either zone by configuring the respective GRABRAM location in the Bank0 USER OTP.
- **Flash Sectors:** Flash sectors of both the banks (Bank0 and Bank1) can be made secure on this device. Each Flash sector can be allocated to either zone by configuring the respective GRABSECT location in the respective bank's USER OTP.
- **Secure ROM:** This device also has secure ROM which is EXEONLY-protected. This ROM contains specific function for the user, provided by TI.

Table 3-15 shows the status of a RAM block/Flash sector based on the configuration in the GRABRAM/GRABSECT register.

**Table 3-15. RAM/Flash Status**

Zone 1 GRAMRAMx/GRABSECTx Bits	Zone 2 GRAMRAMx/GRABSECT	Ownership and Accessibility
01	10	RAM block/Flash Sector belongs to Zone1
01	11 <sup>(1)</sup>	RAM block/Flash Sector belongs to Zone1
10	01	RAM block/Flash Sector belongs to Zone2
11 <sup>(2)</sup>	01	RAM block/Flash Sector belongs to Zone2
10	10	RAM block/Flash Sector is unsecure
11 <sup>(2)</sup>	11 <sup>(1)</sup>	RAM block/Flash Sector is unsecure

<sup>(1)</sup> Zone2 is unsecure. Assumption in this case is that user is not using Zone2 so none of the fields, including passwords, in Zone2 USER OTP are programmed by user hence Zone2 will always be unsecure.

<sup>(2)</sup> Zone1 is unsecure. Assumption in this case is that user is not using Zone1 so none of the fields, including passwords, in Zone1 USER OTP are programmed by user hence Zone1 will always be unsecure.

---

**NOTE:** The user should never program any other values in these fields. Failing any these conditions for a RAM block/Flash sector will make that RAM block/Flash sector inaccessible.

---

The security of each zone is ensured by its own 128-bit (four 32-bit words) password (CSM password). The password for each zone is stored in Bank0 USER OTP. A zone can be unsecured by executing the password match flow (PMF), described in [Section 3.13.7.4](#).



There are three types of accesses: data/program reads, JTAG access, and instruction fetches (calls, jumps, code executions, ISRs). Instruction fetches are never blocked. JTAG accesses are always blocked when a memory is secure. Data reads to a secure memory are always blocked unless the program is executing from a memory which belongs to the same zone. Data reads to unsecure memory are always allowed. [Table 3-16](#) shows the levels of security.

**Table 3-16. Security Levels**

PMF Executed With Correct Password?	Operating Mode of the Zone	Program Fetch Location	Security Description
No	Secure	Outside secure memory	Only instruction fetches by the CPU are allowed to secure memory. In other words, code can still be executed, but not read.
No	Secure	Inside secure memory	CPU has full access (except for EXEONLY memories where read is not allowed). JTAG port cannot read the secured memory contents.
Yes	Unsecure	Anywhere	Full access for CPU and JTAG port to secure memory of that zone.

### 3.13.1.1 CSM Passwords

Unlike earlier C2000 devices, on this device ALL\_1 value (0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF) for CSM password for a zone does not unsecure the zone. Instead, if for any zone the CSM password values get loaded as ALL\_1 from USER OTP, the device will be in BLOCKED state. Due to this reason TI will program a few bits in the second 32-bit password value (ZxOTP\_CSMPSWD1) in every zone select block of each zone with value '0'. The default value for this password location is chosen in a manner that the respective ECC value remains ALL\_1. Due to this, the CSMPSWD1 value programmed by TI for every zone select block is different. Please see [Table 3-17](#) for ZxOTP\_CSMPSWD1 value, programmed by TI on every device. Since ECC is not programmed, the user will be able to change this value by flipping the bits which are '1' to '0' but leaving the ones which are already programmed by TI as '0'. BOOTROM code will write the default password value into the KEYx register to unlock the device as part of device initialization sequence.

If the password locations of a zone have all 128 bits as zeros (ALL\_0), that zone becomes permanently secure (LOCKED state), regardless of the contents of the CSMKEYx registers which means the zone cannot be unlocked using PMF, the password match flow described in [Section 3.13.7.4](#). Therefore, the user should never use ALL\_0 as password. A password of ALL\_0 will prevent debug of secure code or reprogramming the Flash sectors. CSMKEYx registers are user-accessible registers that are used to unsecure the zones.

Default value of other fields in USER OTP are also not ALL\_1. While programming these setting in USER OTP, user must keep already programmed bits '0' only else programming operation will not work.

Default value for all the LINKPOINTER location in USER OTP is 0x1ffffff.

**Table 3-17. Default Value of ZxOTP\_CSMPSWD1 and other fields (programmed by TI)**

Zone Select Block	Zone1		Zone2	
	Address	Value	Address	Value
PSWDLOCK (LSW)	0x00078010	0xfb7ffff	0x00078210	0xbf7ffff
PSWDLOCK (MSW)	0x00078012	0x7ffffff	0x00078212	0x77ffff
CRCLOCK (LSW)	0x00078014	0x4bffff	0x00078214	0x0ffff
CRCLOCK (MSW)	0x00078016	0x3ffff	0x00078216	0x37ffff
Zone_Select_Block0	0x0007802a	0x47ffff	0x0007822a	0xe3ffff
Zone_Select_Block1	0x0007803a	0xdb7ffff	0x0007823a	0x977ffff
Zone_Select_Block2	0x0007804a	0x4bffff	0x0007824a	0xf1ffff

**Table 3-17. Default Value of ZxOTP\_CSMPswd1 and other fields (programmed by TI) (continued)**

Zone Select Block	Zone1		Zone2	
Zone_Select_Block3	0x0007805a	0x3f7ffff	0x0007825a	0x9b7ffff
Zone_Select_Block4	0x0007806a	0xcfbffff	0x0007826a	0x5b7ffff
Zone_Select_Block5	0x0007807a	0x8bffff	0x0007827a	0x2ffff
Zone_Select_Block6	0x0007808a	0x53ffff	0x0007828a	0x1ffff
Zone_Select_Block7	0x0007809a	0xcf7ffff	0x0007829a	0x6b7ffff
Zone_Select_Block8	0x000780aa	0xe77ffff	0x000782aa	0xab7ffff
Zone_Select_Block9	0x000780ba	0x93ffff	0x000782ba	0x37ffff
Zone_Select_Block10	0x000780ca	0xeb7ffff	0x000782ca	0x4f7ffff
Zone_Select_Block11	0x000780da	0x69ffff	0x000782da	0x3bffff
Zone_Select_Block12	0x000780ea	0xa9ffff	0x000782ea	0xe5ffff
Zone_Select_Block13	0x000780fa	0xdd7ffff	0x000782fa	0x8f7ffff
Zone_Select_Block14	0x0007810a	0x8bffff	0x0007830a	0x2ffff
Zone_Select_Block15	0x0007811a	0xcfbffff	0x0007831a	0x5b7ffff
Zone_Select_Block16	0x0007812a	0x3f7ffff	0x0007832a	0x9b7ffff
Zone_Select_Block17	0x0007813a	0x4bffff	0x0007833a	0xf1ffff
Zone_Select_Block18	0x0007814a	0xdb7ffff	0x0007834a	0x977ffff
Zone_Select_Block19	0x0007815a	0x47ffff	0x0007835a	0xe3ffff
Zone_Select_Block20	0x0007816a	0x87ffff	0x0007836a	0xcbffff
Zone_Select_Block21	0x0007817a	0xf37ffff	0x0007837a	0x577ffff
Zone_Select_Block22	0x0007818a	0xdd7ffff	0x0007838a	0x8f7ffff
Zone_Select_Block23	0x0007819a	0xa9ffff	0x0007839a	0xe5ffff
Zone_Select_Block24	0x000781aa	0x69ffff	0x000783aa	0x3bffff
Zone_Select_Block25	0x000781ba	0xeb7ffff	0x000783ba	0x4f7ffff
Zone_Select_Block26	0x000781ca	0x93ffff	0x000783ca	0x37ffff
Zone_Select_Block27	0x000781da	0xe77ffff	0x000783da	0xab7ffff
Zone_Select_Block28	0x000781ea	0xcf7ffff	0x000783ea	0x6b7ffff
Zone_Select_Block29	0x000781fa	0x53ffff	0x000783fa	0x1ffff

### 3.13.1.2 Emulation Code Security Logic (ECSL)

In addition to the CSM, the emulation code security logic (ECSL) has been implemented using a 64-bit password (part of existing CSM password) for each zone to prevent unauthorized users from stepping through secure code. A halt in secure code while the emulator is connected will trip the ECSL and break the emulation connection. To allow emulation of secure code, while maintaining the CSM protection against secure memory reads, the user must write the correct 64-bit password into the CSMKEY (0/1) registers, which matches the password value stored in the USER OTP of that zone. This will disable the ECSL for the specific zone.

When initially debugging a device with the password locations in OTP programmed (secured), the emulator takes some time to take control of the CPU. During this time, the CPU will start running and may execute an instruction that performs an access to a protected ECSL area and if the CPU is halted when the program counter (PC) is pointing to a secure location, the ECSL will trip and cause the emulator connection to be broken.

The solution to this problem is:

- Use the Wait Boot Mode boot option. In this mode, the CPU will be in a loop and hence will not jump to the user application code. Using this BOOTMODE, the user can connect to CCS and debug the code.



### 3.13.1.3 CPU Secure Logic

The CPU Secure Logic (CPUSL) on this device prevents a hacker from reading the CPU registers in a watch window while code is running in a secure zone. All accesses to CPU registers when the PC points to a secure location are blocked by this logic. The only exception to this is read access to the PC. It is highly recommended not to write into the CPU register in this case, because proper code execution may get affected. If the CSM is unlocked using the CSM password match flow, the CPUSL logic also gets disabled.

### 3.13.1.4 Execute-Only Protection

To achieve a higher level of security on secure Flash sectors and RAM blocks that store critical user code (instruction opcodes), the Execute-Only protection feature is provided. When the Execute-Only protection is turned on for any secure Flash sector or RAM block, data reads to that Flash sector or RAM block are disallowed from any code (even from secure code). Execute-only protection for a Flash sector and RAM block can be turned on by configuring the bit field associated for that particular sector/RAM block in the zone's (which has ownership of that sector/RAM block) EXEONLYSECT and EXEONLYRAM register, respectively.

### 3.13.1.5 Password Lock

The password locations in USER OTP for for each zone can be locked by programming the zone's PSWDLOCK field with any value other than "1111" (0xF) at the PSWDLOCK location in OTP. Until the passwords of a zone are locked, password locations will not be secure and can be read from the debugger as well as code running from non-secure memory. This feature can be used by the user to avoid accidental locking of the zone while programming the Flash sectors during the software development phase. On a fresh device the value for password lock fields for all zones at the PSWDLOCK location in OTP will be "1111" which means the password for all zones will be unlocked.

---

**NOTE:** Password unlock only makes password locations non-secure. All other secure memories and other locations of Flash sectors, which contain a password, remains secure as per security settings. But since passwords are non-secure, anyone can read the password and make the zone non-secure by running through PMF.

---

### 3.13.1.6 Link Pointer and Zone Select

For each of the two security zones, a dedicated OTP block exists in both the banks which holds the configuration related to the zone's security. The following are the available programmed configurations:

#### **BANK0 USER OTP**

- B0\_ZxOTP\_LINKPOINTER1
- B0\_ZxOTP\_LINKPOINTER2
- B0\_ZxOTP\_LINKPOINTER3
- ZxOTP\_GPREG1
- ZxOTP\_GPREG2
- ZxOTP\_PSWDLOCK
- ZxOTP\_CRCLOCK
- ZxOTP\_BOOTCTRL
- ZxOTP\_GPREG3
- ZxOTP\_EXEONLYRAM
- B0\_ZxOTP\_EXEONLYSECT
- ZxOTP\_GRABRAM
- B0\_ZxOTP\_GRABSECT
- ZxOTP\_CSMPSWD0
- ZxOTP\_CSMPSWD1

- ZxOTP\_CSMPSWD2
- ZxOTP\_CSMPSWD3

**BANK1 USER OTP**

- B1\_ZxOTP\_LINKPOINTER1
- B1\_ZxOTP\_LINKPOINTER2
- B1\_ZxOTP\_LINKPOINTER3
- B1\_ZxOTP\_EXEONLYSECT
- B1\_ZxOTP\_GRABSECT

Since OTP cannot be erased, the following configurations are placed in zone select blocks of each zone's OTP Flash of both the banks.

- ZxOTP\_EXEONLYRAM
- B0\_ZxOTP\_EXEONLYSECT
- ZxOTP\_GRABRAM
- B0\_ZxOTP\_GRABSECT
- ZxOTP\_CSMPSWD0
- ZxOTP\_CSMPSWD1
- ZxOTP\_CSMPSWD2
- ZxOTP\_CSMPSWD3
- B1\_ZxOTP\_EXEONLYSECT
- B1\_ZxOTP\_GRABSECT

The location of the zone select block in OTP is decided based on the value of three 29-bit link pointers (Bx-Zx-LINKPOINTERx) programmed in the OTP of each zone. All OTP locations except link pointer locations are protected with ECC. Since the link pointer locations are not protected with ECC, three link pointers are provided that need to be programmed with the same value. The final value of the link pointer is resolved in hardware when a dummy read is done to all the link pointers by comparing all the three values (bit-wise voting logic). Since in OTP, a '1' can be flipped by the user to '0' but '0' can't be flipped to '1' (no erase operation for OTP), the most significant bit position in the resolved link pointer which is '0', defines the valid base address for the zone select block. While generating the final link pointer value, if the bit pattern is not one of those listed in [Figure 3-18](#), the final link pointer value becomes All\_1 (0xFFFF\_FFFF) which selects the Zone-Select-Block1 (also known as the default zone select block).

**Figure 3-18. Storage of Zone-Select Bits in OTP**

Zx-LINKPOINTER	Addr Offset Of Zone-Select Block
32'b00011111111111111111111111111111	0x20
32'b00011111111111111111111111111111 0	0x30
32'b00011111111111111111111111111111 0x	0x40
32'b00011111111111111111111111111111 0xx	0x50
32'b00011111111111111111111111111111 0xxx	0x60
32'b00011111111111111111111111111111 0xxxx	0x70
32'b00011111111111111111111111111111 0xxxxx	0x80
32'b00011111111111111111111111111111 0xxxxxx	0x90
32'b00011111111111111111111111111111 0xxxxxxx	0xa0
32'b00011111111111111111111111111111 0xxxxxxxx	0xb0
32'b00011111111111111111111111111111 0xxxxxxxxx	0xc0
32'b00011111111111111111111111111111 0xxxxxxxxxx	0xd0
32'b00011111111111111111111111111111 0xxxxxxxxxxx	0xe0
32'b00011111111111111111111111111111 0xxxxxxxxxxxx	0xf0
32'b00011111111111111111111111111111 0xxxxxxxxxxxxx	0x100
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxx	0x110
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x120
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x130
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x140
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x150
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x160
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x170
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x180
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x190
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x1a0
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x1b0
32'b00011111111111111111111111111111 0xxxxxxxxxxxxxxx	0x1c0
32'b000110xxxxxxxxxxxxxxxxxxxxxxxxxxxxx	0x1d0
32'b00010xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	0x1e0
32'b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	0x1f0

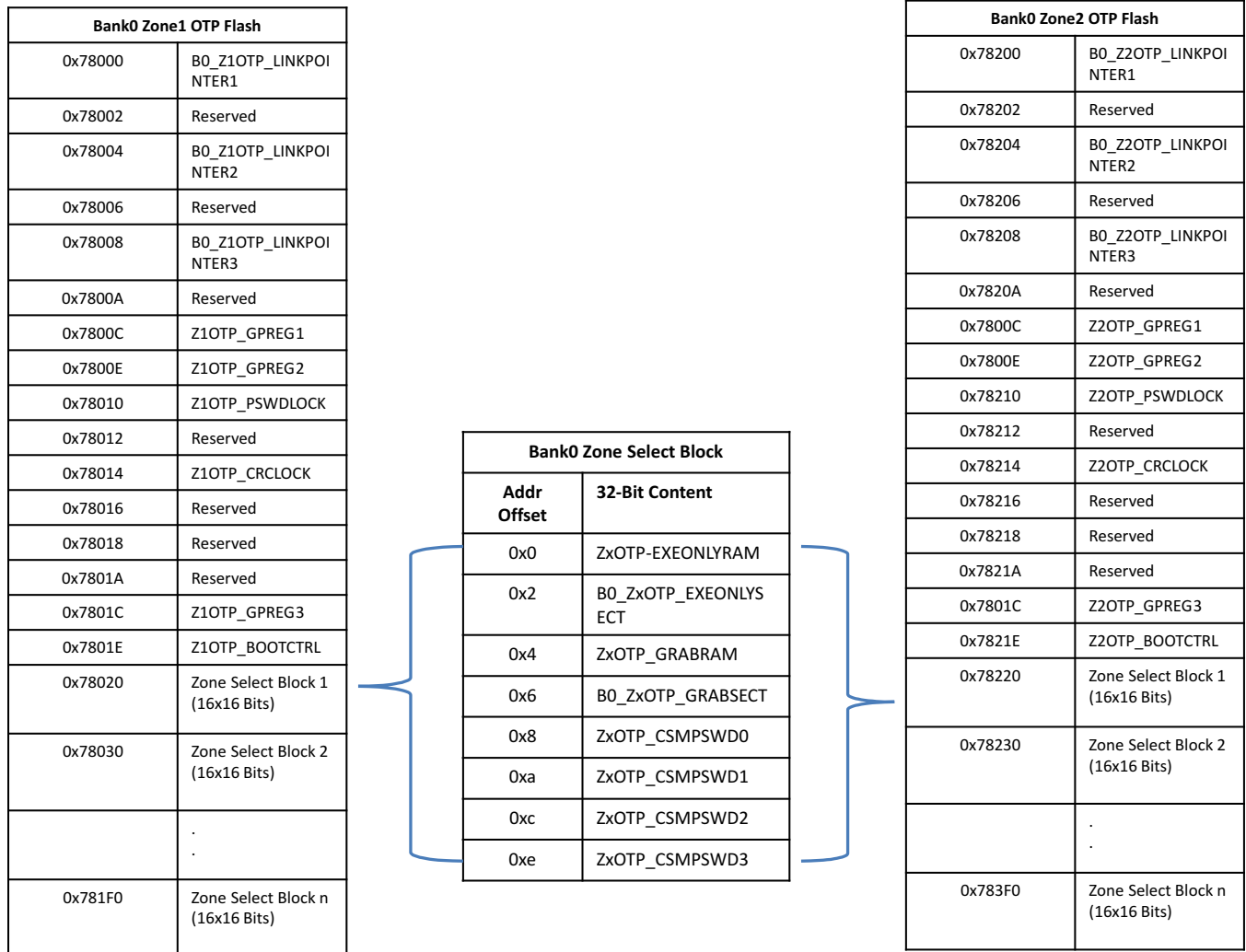
  

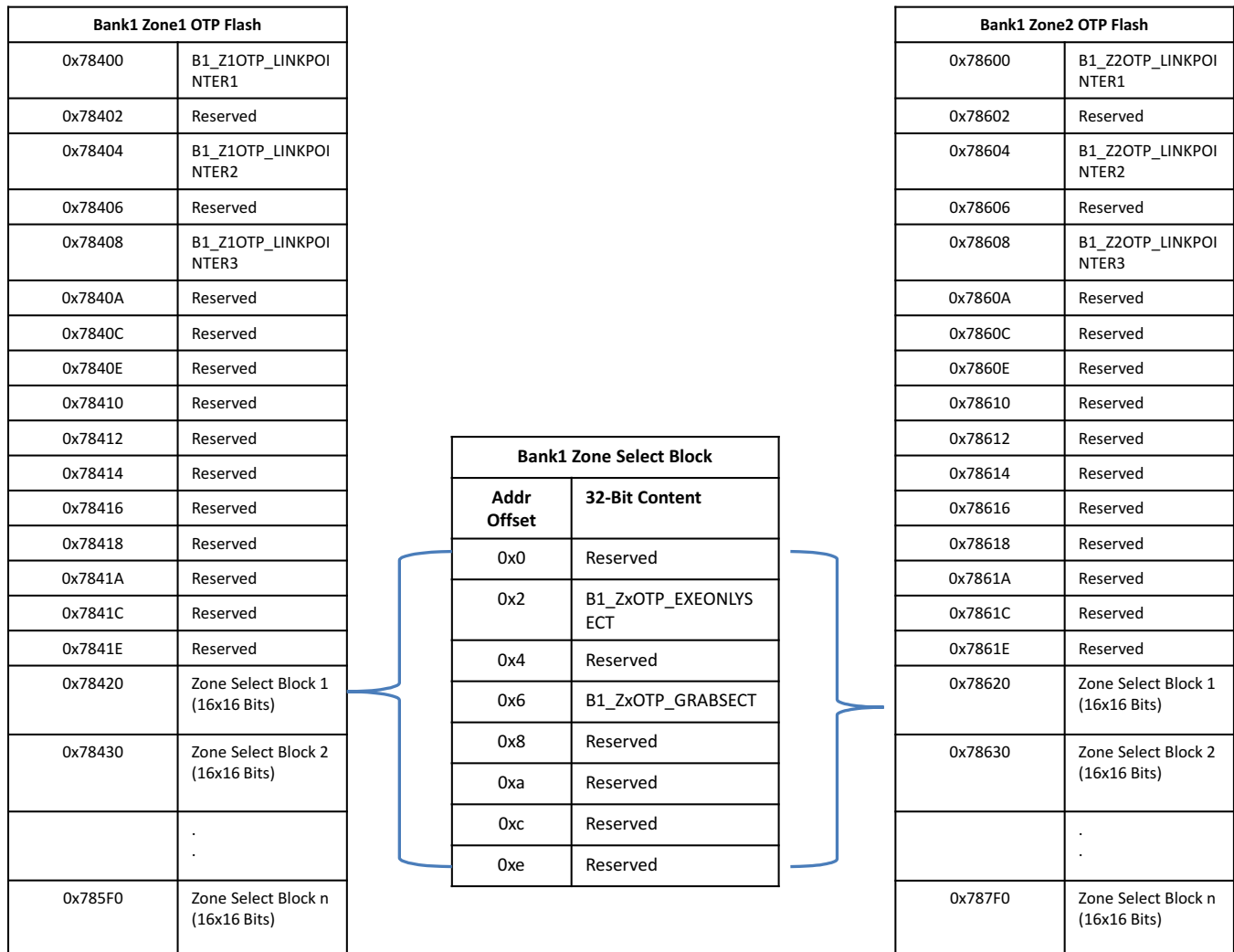
Zone Select Block	
Addr Offset	32-Bit Content
0x0	Zx-EXEONLYRAM
0x2	Bx-Zx EXEONLYSECT
0x4	Zx-GRABRAM
0x6	Bx-Zx GRABSECT
0x8	Zx-CSMPSWD0
0xa	Zx-CSMPSWD1
0xc	Zx-CSMPSWD2
0xe	Zx-CSMPSWD3

**NOTE:** Address locations for other security settings (PSWDLOCK/CRCLOCK) that are not part of Zone Select blocks) can be programmed only once; therefore, the user should program them towards end of the development cycle.

**NOTE:** Zone select blocks in BANK1 only have Zx-EXEONLYSECT and Zx-GRABSECT as valid configurations. Other locations in zone select block of BANK1 are reserved.

**Figure 3-19. Location of Zone-Select Block Based on Link-Pointer for Bank0**



**Figure 3-20. Location of Zone-Select Block Based on Link-Pointer for Bank1**


**NOTE:** USER OTP is ECC protected. The user must program the ECC value while programming the security setting in USER OTP. Failing to program the correct ECC value may cause the device to be blocked permanently and the user will have to replace the device.

### 3.13.2 C Code Example to get Zone Select Block Addr for Zone1 in BANK0

```

unsigned long LinkPointer;
unsigned long *Zone1SelBlockPtr;
int Bitpos = 28;
int ZeroFound = 0;

// Read Z1-Linkpointer register of DCSM module.
LinkPointer = *(unsigned long *)0x5F000;

// Bits 31 30 and 29 as most-significant 0 are reserved LinkPointer options

LinkPointer = LinkPointer << 3;

while ((ZeroFound == 0) && (bitpos > -1))
{
    if ((LinkPointer & 0x80000000) == 0)
    {
        ZeroFound = 1;
        Zone1SelBlockPtr = (unsigned long *) (0x78000 + ((bitpos + 3)*16));
    } else
    {
        bitpos--;
        LinkPointer = LinkPointer << 1;
    }
}
if (ZeroFound == 0)
{
    //Default in case there is no zero found.
    Zone1SelBlockPtr = (unsigned long *)0x78020;
}

```

### 3.13.3 Flash and OTP Erase/Program

On this device, OTP as well as normal Flash, are secure resources. Each zone has its own dedicated OTP, whereas normal Flash sectors can be allocated to any zone based on the value programmed in the GRABSECT location in OTP of both the banks. Each zone has its own CSM passwords; read and write accesses are not allowed to resources assigned to Z1 by code running from memory allocated to Zone 2 and vice versa. Before programming any secure Flash sector, the user must either unlock the zone to which that particular sector belongs, using PMF or execute the Flash programming code from secure memory which belongs to the same zone. The same is the case for erasing any secure Flash sector. To program the security settings in OTP Flash, the user must unlock the CSM of the respective zone. Unless the zone is unlocked, security settings in OTP Flash can not be updated. The OTP content cannot be erased.

The flash wrapper registers through which any flash or OTP program/erase operations are performed are common resources shared by the two security zones. A semaphore mechanism is provided to avoid the conflict between Zone1 and Zone2. A zone needs to grab this semaphore to successfully complete the erase/program operation on the Flash sectors allocated to that zone. A semaphore can be grabbed by a zone by writing the appropriate value in the SEM field of the FLSEM register. For further details of this field, see the register description.

### 3.13.4 Safe Copy Code

In some applications, the user may want to copy the code from secure Flash to secure RAM for better performance. The user cannot do this for EXEONLY flash sectors because EXEONLY secure memories cannot be read from anywhere. TI provides specific "Safe Copy Code" library functions for each zone to enable the user to copy content from EXEONLY secure flash sectors to EXEONLY RAM blocks. These functions do the copy-code operation in a highly secure environment and allow a copy to be performed only when the following conditions are met:

- The secure RAM block and the secure flash sector belong to the same zone.
- Both the secure RAM block and the secure flash sector have EXEONLY protection enabled.

For further usage of these library functions, see the device-specific Boot ROM documentation.

### 3.13.5 SafeCRC

Since reads from EXEONLY memories are not allowed, the user cannot calculate the CRC for content in EXEONLY memories using the VCU. But in some safety-critical applications, the user may have to calculate the CRC on these memories as well. To enable this without compromising on security, TI provides specific “SafeCRC” library functions for each zone. These functions do the CRC calculation in highly secure environment and allow a CRC calculation to be performed only when the following conditions are met:

- The source address should be modulo the number of words (based on length\_id) for which the CRC needs to be calculated.
- The destination address should belong to the same zone as the source address.

For further usage of these library functions, see the device-specific Boot ROM documentation.

---

**NOTE:** The user must disable all the interrupts before calling the safe copy code and the safeCRC function. If there is a vector fetch during copy code operation, the CPU gets reset immediately.

---

Disclaimer: The Code Security Module (CSM) included on this device was designed to password protect the data stored in the associated memory and is warranted by Texas Instruments (TI), in accordance with its standard terms and conditions, to conform to TI's published specifications for the warranty period applicable for this device. TI DOES NOT, HOWEVER, WARRANT OR REPRESENT THAT THE CSM CANNOT BE COMPROMISED OR BREACHED OR THAT THE DATA STORED IN THE ASSOCIATED MEMORY CANNOT BE ACCESSED THROUGH OTHER MEANS. MOREOVER, EXCEPT AS SET FORTH ABOVE, TI MAKES NO WARRANTIES OR REPRESENTATIONS CONCERNING THE CSM OR OPERATION OF THIS DEVICE, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL TI BE LIABLE FOR ANY CONSEQUENTIAL, SPECIAL, INDIRECT, INCIDENTAL, OR PUNITIVE DAMAGES, HOWEVER CAUSED, ARISING IN ANY WAY OUT OF YOUR USE OF THE CSM OR THIS DEVICE, WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO LOSS OF DATA, LOSS OF GOODWILL, LOSS OF USE OR INTERRUPTION OF BUSINESS OR OTHER ECONOMIC LOSS.

### 3.13.6 CSM Impact on Other On-Chip Resources

On this device, M0/M1 and GSx memories are not secure. To avoid any potential hacking when the device is in the default state (post reset), accesses (all types) to all memories (secure as well as non-secure, except BOOT-ROM and OTP ) are disabled until proper security initialization is done. This means that after reset none of the memory resources except BOOT\_ROM and OTP is accessible to the user.

The following steps are required after reset (any type of reset) to initialize the security on each CPU subsystem.

#### Security Initialization

- Dummy Read to address location of B0\_Z1OTP\_LINKPOINTER1 in Z1 OTP of Bank0
- Dummy Read to address location of B0\_Z1OTP\_LINKPOINTER2 in Z1 OTP of Bank0
- Dummy Read to address location of B0\_Z1OTP\_LINKPOINTER3 in Z1 OTP of Bank0
- Dummy Read to address location of B0\_Z2OTP\_LINKPOINTER1 in Z2 OTP of Bank0
- Dummy Read to address location of B0\_Z2OTP\_LINKPOINTER2 in Z2 OTP of Bank0
- Dummy Read to address location of B0\_Z2OTP\_LINKPOINTER3 in Z2 OTP of Bank0
- Dummy Read to address location of B1\_Z1OTP\_LINKPOINTER1 in Z1 OTP of Bank1
- Dummy Read to address location of B1\_Z1OTP\_LINKPOINTER2 in Z1 OTP of Bank1
- Dummy Read to address location of B1\_Z1OTP\_LINKPOINTER3 in Z1 OTP of Bank1
- Dummy Read to address location of B1\_Z2OTP\_LINKPOINTER1 in Z2 OTP of Bank1
- Dummy Read to address location of B1\_Z2OTP\_LINKPOINTER2 in Z2 OTP of Bank1



- Dummy Read to address location of B1\_Z2OTP\_LINKPOINTER3 in Z2 OTP of Bank1
- Dummy Read to address location of SECDC (0x703F0, TI-reserved register) in TI OTP
- Dummy Read to address location of Z1OTP\_PSWDLOCK in Z1 OTP
- Dummy Read to address location of Z1OTP\_CRCLOCK in Z1 OTP
- Dummy Read to address location 0x78018 in Z1 OTP
- Dummy Read to address location of Z1OTP\_GPREG1, Z1OTP\_GPREG2, Z1OTP\_GPREG3 in Z1 OTP of Bank0
- Dummy Read to address location of Z1OTP\_BOOTCTRL in Z1 OTP
- Dummy Read to address location of Z2OTP\_PSWDLOCK in Z2 OTP of Bank0
- Dummy Read to address location of Z2OTP\_CRCLOCK in Z2 OTP of Bank0
- Dummy Read to address location 0x78218 in Z2 OTP of Bank0
- Dummy Read to address location of Z2OTP\_GPREG1, Z2OTP\_GPREG2, Z2OTP\_GPREG3 in Z2 OTP of Bank0
- Dummy Read to address location of Z2OTP\_BOOTCTRL in Z2 OTP of Bank0
- Read to memory map register of B0\_Z1\_LINKPOINTER in DCSM module to calculate the address of zone select block for Z1
- Dummy read to address location of Z1OTP\_EXEONLYRAM in Z1 OTP of Bank0
- Dummy read to address location of B0\_Z1OTP\_EXEONLYSECT in Z1 OTP of Bank0
- Dummy read to address location of Z1OTP\_GRABRAM in Z1 OTP of Bank0
- Dummy read to address location of B0\_Z1OTP\_GRABSECT in Z1 OTP of Bank0
- Read to memory map register of B0\_Z2\_LINKPOINTER in DCSM module to calculate the address of zone select block for Z2
- Dummy read to address location of Z2OTP\_EXEONLYRAM in Z2 OTP of Bank0
- Dummy read to address location of B0\_Z2OTP\_EXEONLYSECT in Z2 OTP of Bank0
- Dummy read to address location of Z2OTP\_GRABRAM in Z2 OTP of Bank0
- Dummy read to address location of B0\_Z2OTP\_GRABSECT in Z2 OTP of Bank0
- Read to memory map register of B1\_Z1\_LINKPOINTER in DCSM module to calculate the address of zone select block for Z1
- Dummy read to address location of B1\_Z1OTP\_EXEONLYSECT in Z1 OTP of Bank1
- Dummy read to address location of B1\_Z1OTP\_GRABSECT in Z1 OTP of Bank1
- Read to memory map register of B1\_Z2\_LINKPOINTER in DCSM module to calculate the address of zone select block for Z2
- Dummy read to address location of B1\_Z2OTP\_EXEONLYSECT in Z2 OTP of Bank1
- Dummy read to address location of B1\_Z2OTP\_GRABSECT in Z2 OTP of Bank1

---

**NOTE:** Security Initialization is done by BOOTROM code on all the resets (as part of device initialization). This will not be part of user application code

---



---

**NOTE:** The order of initialization matters hence if a memory watch window with the USER OTP address is opened in the debugger (CCS) the security initialization could occur in an incorrect order, locking the device down. To avoid this, user should not keep a memory window with USER OTP address opened in the debugger(CCS) when performing a reset.

---

### 3.13.7 Incorporating Code Security in User Applications

Code security is typically not required in the development phase of a project. However, security is needed once a robust code is developed for a zone. Before such a code is programmed in the Flash memory, a CSM password should be chosen to secure the zone. Once a CSM password is in place for a zone, the zone is secured (programming a password at the appropriate locations and either performing a device reset or setting the FORCESEC bit (Zx\_CR.15) is the action that secures the device). From that time on, access to debug the contents of secure memory by any means (via JTAG, code running off external/on-chip memory, and so forth) requires a valid password. A password is not needed to run the code out of secure memory (such as in a typical end-user usage); however, access to secure memory contents for debug purposes, requires a password.

#### 3.13.7.1 Environments That Require Security Unlocking

The following are the typical situations under which unsecuring can be required:

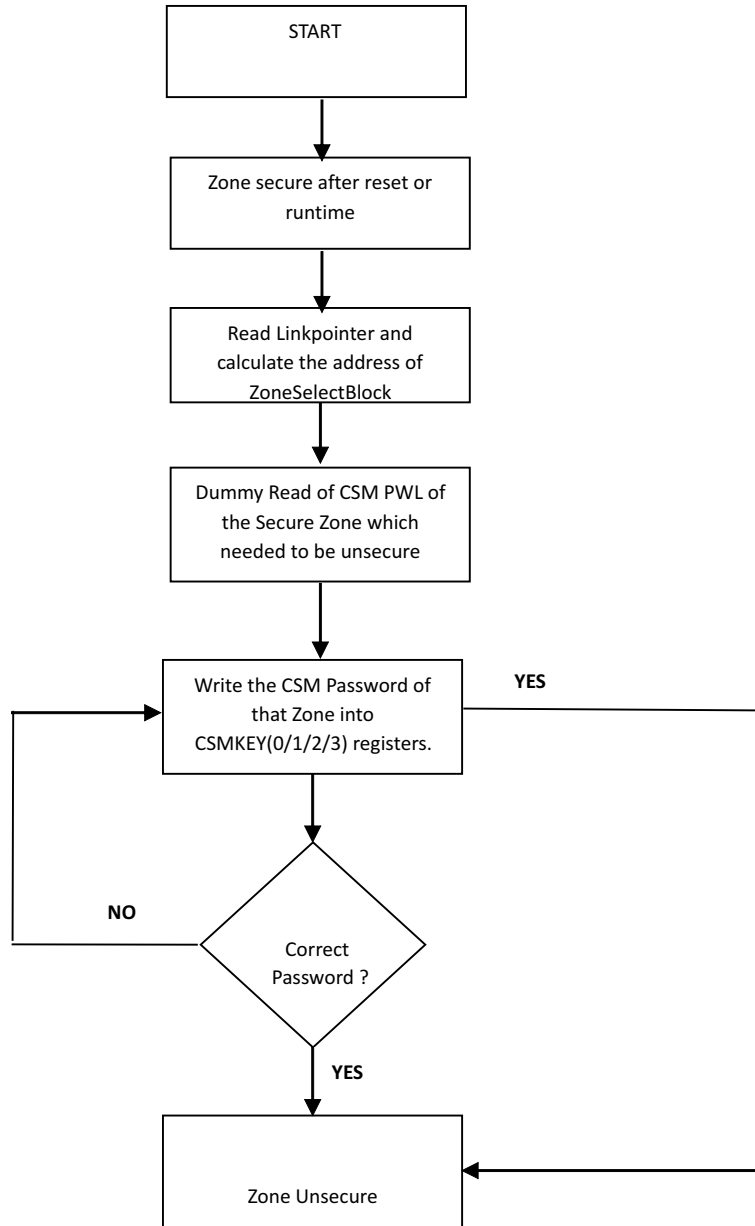
- Code development using debuggers (such as Code Composer Studio). This is the most common environment during the design phase of a product.
- Flash programming using TI's Flash utilities such as Code Composer Studio On-Chip Flash Programmer plug-in or the Uniflash tool. Flash programming is common during code development and testing. Once the user supplies the necessary password, the flash utilities disable the security logic before attempting to program the Flash. The flash utilities can disable the code security logic in new devices without any authorization, since new devices come with an erased Flash. However, reprogramming devices that already contain a custom password require the password to be supplied to the flash utilities in order to unlock the device to enable programming. In custom programming solutions that use the Flash API supplied by TI, unlocking the CSM can be avoided by executing the Flash programming algorithms from secure memory.
- Custom environment defined by the application  
In addition to the above, access to secure memory contents can be required in situations such as:
  - Using the on-chip bootloader to load code or data into secure SARAM or to erase and program the Flash.
  - Executing code from on-chip unsecure memory and requiring access to secure memory for the lookup table. This is not a suggested operating condition as supplying the password from external code could compromise code security.

The unsecuring sequence is identical in all the above situations. This sequence is referred to as the password match flow (PMF) for simplicity. [Figure 3-21](#) explains the sequence of operation that is required every time the user attempts to unsecure a particular zone. A code example is listed for clarity.

**3.13.7.2 CSM Password Match Flow**

Password match flow (PMF) is essentially a sequence of four dummy reads from password locations (PWL) followed by four writes (32-bit writes) to CSMKEY(0/1/2/3) registers. [Figure 3-21](#) shows how PMF helps to initialize the security logic registers and disable security logic.

**Figure 3-21. CSM Password Match Flow (PMF)**



### 3.13.7.3 C Code Example to Unsecure C28x Zone1

```

volatile long int *CSM = (volatile long int *)0x5F000; //CSM register file
volatile long int *CSMPWL = (volatile long int *)0x78028; //CSM Password location (assuming
default Zone select block)

                                                                    volatile int tmp;

int I;
// Read the 128-bits of the CSM password locations (PWL)
//
for (I=0;I<4; I++) tmp = *CSMPWL++;
// If the password locations (CSMPWL) are all = ones (0xFFFF),
// then the zone will now be unsecure. If the password
// is not all ones (0xFFFF), then the code below is required
// to unsecure the CSM.
// Write the 128-bit password to the CSMKEY registers
// If this password matches that stored in the
// CSLPWL then the CSM will become unsecure. If it does not
// match, then the zone will remain secure.
// An example password of:
// 0x11112222333344445555666677778888 is used.
*CSM++ = 0x22221111; // Register Z1_CSMKEY0 at 0x5F010
*CSM++ = 0x44443333; // Register Z1_CSMKEY1 at 0x5F012
*CSM++ = 0x66665555; // Register Z1_CSMKEY2 at 0x5F014
*CSM++ = 0x88887777; // Register Z1_CSMKEY3 at 0x5F016
    
```

### 3.13.7.4 C Code Example to Resecure C28x Zone1

```

volatile int *Z1_CR = 0x5F019; //CSMSCR register
//Set FORCESEC bit
*Z1_CR = 0x8000;
    
```

### 3.13.7.5 Environments That Require ECSL Unlocking

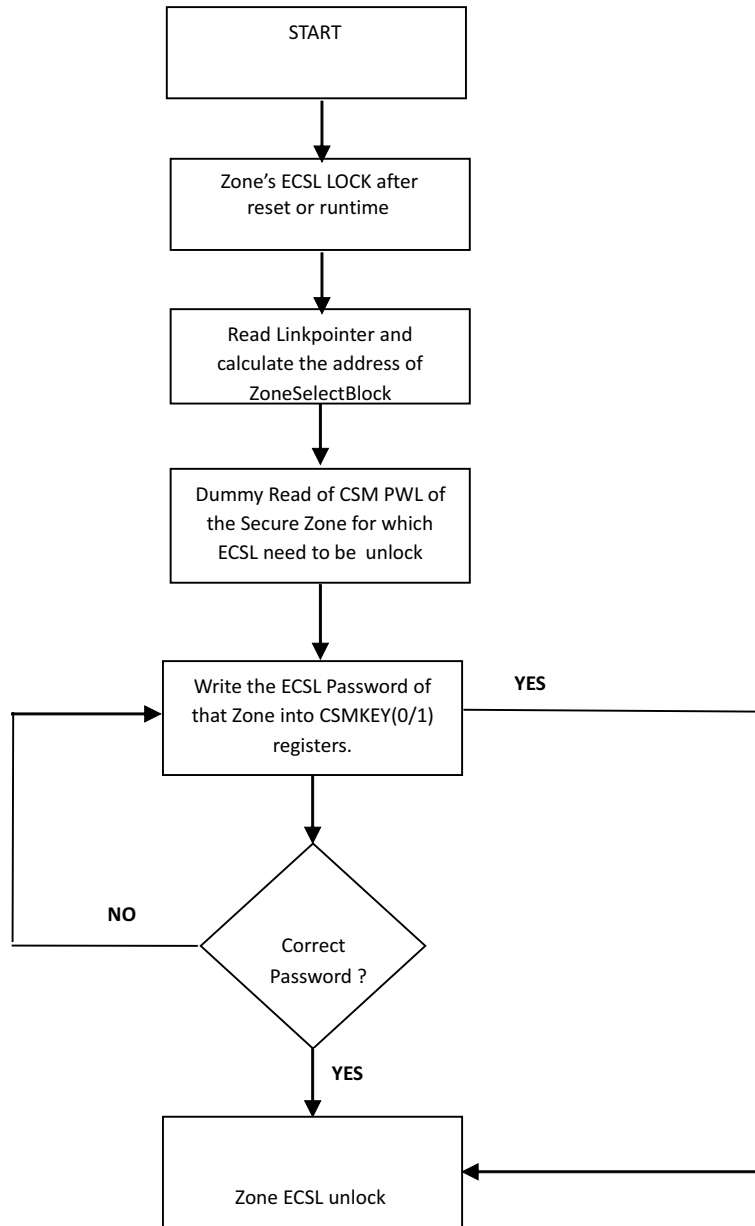
The following are the typical situations under which unsecuring can be required:

- The user develops some main IP, and then outsources peripheral functions to a subcontractor who must be able to run the user code during debug and may halt while main IP code is running. If ECSL is not unlocked, then Code Composer Studio connections will get disconnected, which can be inconvenient for the user. Note that unlocking ECSL doesn't enable access to secure code but only avoids disconnection of CCS (JTAG).

### 3.13.7.6 ECSL Password Match Flow

A password match flow (PMF) is essentially a sequence of eight dummy reads from password locations (PWL) followed by two writes to KEY registers. [Figure 3-22](#) shows how the PMF helps to initialize the security logic registers and disable security logic.

Figure 3-22. ECSL Password Match Flow (PMF)



### 3.13.7.7 ECSL Disable Considerations for any Zone

A zone with ECSL enabled should have a predetermined ECSL password stored in the ECSL password locations in Flash (same as lower 64 bits of CSM passwords). The following are steps to disable the ECSL for any particular zone:

- Perform a dummy read of CSM password locations of that Zone.
- Write the password into the CSMKEY0/1 registers, corresponding to that Zone.
- If the password is correct, the ECSL gets disabled; otherwise, it stays enabled.

### 3.13.7.7.1 C Code Example to Disable ECSL for C28x-Zone1

```

volatile long int *ECSL = (volatile int *)0x5F000; //ECSL register file
volatile long int *ECSLPWL = (volatile int *)0x78028; //ECSL Password location (assuming default
Zone sel block)

                                                                    volatile int tmp;

int I;
// Read the 64-bits of the password locations (PWL)
.
for (I=0;I<2; I++) tmp = *ECSLPWL++;
// If the ECSL password locations (ECSLPWL) are all = ones (0xFFFF),
// then the ECSL will now be disable. If the password
// is not all ones (0xFFFF), then the code below is required
// to disable the ECSL.
// Write the 64-bit password to the CSMKEYx registers
// If this password matches that stored in the
// CSMPWL then ECSL will get disable. If it does not
// match, then the zone will remain secure.
// An example password of:
// 0x1111222233334444 is used.
*ECSL++ = 0x22221111; // Register Z1_CSMKEY0 at 0x5F010
*ECSL++ = 0x44443333; // Register Z1_CSMKEY1 at 0x5F012

```

### 3.13.7.8 Device Unique ID

Bank 0 OTP contains a 256-bit value that is made up of both random and sequential parts. This value can be used as a seed for code encryption. The starting address of the value is 0x703C0. The first 192 bits are random, the next 32 bits are sequential, and the last 32 bits are a checksum value.

## 3.14 F28004x System Control Registers

### 3.14.1 System Control Base Addresses

**Table 3-18. System Control Base Address Table**

Device Registers	Register Name	Start Address	End Address
CpuTimer0Regs	CPUTIMER_REGS	0x0000_0C00	0x0000_0C07
CpuTimer1Regs	CPUTIMER_REGS	0x0000_0C08	0x0000_0C0F
CpuTimer2Regs	CPUTIMER_REGS	0x0000_0C10	0x0000_0C17
PieCtrlRegs	PIE_CTRL_REGS	0x0000_0CE0	0x0000_0CFF
WdRegs	WD_REGS	0x0000_7000	0x0000_703F
NmiIntruptRegs	NMI_INTRUPT_REGS	0x0000_7060	0x0000_706F
XintRegs	XINT_REGS	0x0000_7070	0x0000_707F
DmaClaSrcSelRegs	DMA_CLA_SRC_SEL_REGS	0x0000_7980	0x0000_79BF
DevCfgRegs	DEV_CFG_REGS	0x0005_D000	0x0005_D17F
ClkCfgRegs	CLK_CFG_REGS	0x0005_D200	0x0005_D2FF
CpuSysRegs	CPU_SYS_REGS	0x0005_D300	0x0005_D3FF
PeriphAcRegs	PERIPH_AC_REGS	0x0005_D500	0x0005_D6FF
RomPrefetchRegs	ROM_PREFETCH_REGS	0x0005_E608	0x0005_E609
DccRegs	DCC_REGS	0x0005_E700	0x0005_E73F
DcsmBank0Z1Regs	DCSM_BANK0_Z1_REGS	0x0005_F000	0x0005_F02F
DcsmBank0Z2Regs	DCSM_BANK0_Z2_REGS	0x0005_F040	0x0005_F06F
DcsmBank1Z1Regs	DCSM_BANK1_Z1_REGS	0x0005_F100	0x0005_F12F
DcsmBank1Z2Regs	DCSM_BANK1_Z2_REGS	0x0005_F140	0x0005_F16F
DcsmCommonRegs	DCSM_COMMON_REGS	0x0005_F070	0x0005_F07F
DcsmCommon2Regs	DCSM_COMMON2_REGS	0x0005_F080	0x0005_F087
MemCfgRegs	MEM_CFG_REGS	0x0005_F400	0x0005_F47F
AccessProtectionRegs	ACCESS_PROTECTION_REGS	0x0005_F4C0	0x0005_F4FF
MemoryErrorRegs	MEMORY_ERROR_REGS	0x0005_F500	0x0005_F53F
RomWaitStateRegs	ROM_WAIT_STATE_REGS	0x0005_F540	0x0005_F541
Flash0CtrlRegs	FLASH_CTRL_REGS	0x0005_F800	0x0005_FAFF
Flash0EccRegs	FLASH_ECC_REGS	0x0005_FB00	0x0005_FB3F
UidRegs	UID_REGS	0x0007_03C0	0x0007_03CF



### 3.14.2 ACCESS\_PROTECTION\_REGS Registers

Table 3-19 lists the ACCESS\_PROTECTION\_REGS registers. All register offset addresses not listed in Table 3-19 should be considered as reserved locations and the register contents should not be modified.

**Table 3-19. ACCESS\_PROTECTION\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	NMAVFLG	Non-Master Access Violation Flag Register		<a href="#">Go</a>
2h	NMAVSET	Non-Master Access Violation Flag Set Register	EALLOW	<a href="#">Go</a>
4h	NMAVCLR	Non-Master Access Violation Flag Clear Register	EALLOW	<a href="#">Go</a>
6h	NMAVINTEN	Non-Master Access Violation Interrupt Enable Register	EALLOW	<a href="#">Go</a>
8h	NMCPURDAVADDR	Non-Master CPU Read Access Violation Address		<a href="#">Go</a>
Ah	NMCPUWRAVADDR	Non-Master CPU Write Access Violation Address		<a href="#">Go</a>
Ch	NMCPUFAVADDR	Non-Master CPU Fetch Access Violation Address		<a href="#">Go</a>
Eh	NMDMAWRAVADDR	Non-Master DMA Write Access Violation Address		<a href="#">Go</a>
10h	NMCLA1RDAVADDR	Non-Master CLA1 Read Access Violation Address		<a href="#">Go</a>
12h	NMCLA1WRAVADDR	Non-Master CLA1 Write Access Violation Address		<a href="#">Go</a>
14h	NMCLA1FAVADDR	Non-Master CLA1 Fetch Access Violation Address		<a href="#">Go</a>
20h	MAVFLG	Master Access Violation Flag Register		<a href="#">Go</a>
22h	MAVSET	Master Access Violation Flag Set Register	EALLOW	<a href="#">Go</a>
24h	MAVCLR	Master Access Violation Flag Clear Register	EALLOW	<a href="#">Go</a>
26h	MAVINTEN	Master Access Violation Interrupt Enable Register	EALLOW	<a href="#">Go</a>
28h	MCPUFAVADDR	Master CPU Fetch Access Violation Address		<a href="#">Go</a>
2Ah	MCPUWRAVADDR	Master CPU Write Access Violation Address		<a href="#">Go</a>
2Ch	MDMAWRAVADDR	Master DMA Write Access Violation Address		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-20 shows the codes that are used for access types in this section.

**Table 3-20. ACCESS\_PROTECTION\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		

**Table 3-20. ACCESS\_PROTECTION\_REGS Access  
Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.2.1 NMAVFLG Register (Offset = 0h) [reset = 0h]

NMAVFLG is shown in [Figure 3-23](#) and described in [Table 3-21](#).

Return to the [Summary Table](#).

Non-Master Access Violation Flag Register

**Figure 3-23. NMAVFLG 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						RESERVED	RESERVED
R-0h							
7	6	5	4	3	2	1	0
RESERVED	CLA1FETCH	CLA1WRITE	CLA1READ	DMAWRITE	CPUFETCH	CPUWRITE	CPUREAD
	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-21. NMAVFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	CLA1FETCH	R	0h	Non Master CLA1 Fetch Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn
5	CLA1WRITE	R	0h	Non Master CLA1 Write Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn
4	CLA1READ	R	0h	Non Master CLA1 Read Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn
3	DMAWRITE	R	0h	Non Master DMA Write Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn
2	CPUFETCH	R	0h	Non Master CPU Fetch Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn

**Table 3-21. NMAVFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	CPUWRITE	R	0h	Non Master CPU Write Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn
0	CPUREAD	R	0h	Non Master CPU Read Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn

**3.14.2.2 NMAVSET Register (Offset = 2h) [reset = 0h]**

NMAVSET is shown in [Figure 3-24](#) and described in [Table 3-22](#).

Return to the [Summary Table](#).

Non-Master Access Violation Flag Set Register

**Figure 3-24. NMAVSET 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						RESERVED	RESERVED
R-0h							
7	6	5	4	3	2	1	0
RESERVED	CLA1FETCH	CLA1WRITE	CLA1READ	DMAWRITE	CPUFETCH	CPUWRITE	CPUREAD
	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-22. NMAVSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-10	RESERVED	R	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	RESERVED	R-0/W1S	0h	Reserved
6	CLA1FETCH	R-0/W1S	0h	0: No action. 1: CLA1 Fetch Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn
5	CLA1WRITE	R-0/W1S	0h	0: No action. 1: CLA1 Write Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn
4	CLA1READ	R-0/W1S	0h	0: No action. 1: CLA1 Read Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn
3	DMAWRITE	R-0/W1S	0h	0: No action. 1: DMA Write Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn
2	CPUFETCH	R-0/W1S	0h	0: No action. 1: CPU Fetch Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn
1	CPUWRITE	R-0/W1S	0h	0: No action. 1: CPU Write Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn

**Table 3-22. NMAVSET Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	CPUREAD	R-0/W1S	0h	0: No action. 1: CPU Read Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn

### 3.14.2.3 NMAVCLR Register (Offset = 4h) [reset = 0h]

NMAVCLR is shown in [Figure 3-25](#) and described in [Table 3-23](#).

Return to the [Summary Table](#).

Non-Master Access Violation Flag Clear Register

**Figure 3-25. NMAVCLR 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						RESERVED	RESERVED
R-0h							
7	6	5	4	3	2	1	0
RESERVED	CLA1FETCH	CLA1WRITE	CLA1READ	DMAWRITE	CPUFETCH	CPUWRITE	CPUREAD
	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-23. NMAVCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-10	RESERVED	R	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	RESERVED	R-0/W1S	0h	Reserved
6	CLA1FETCH	R-0/W1S	0h	0: No action. 1: CLA1 Fetch Access Violation Flag in NMAVFLG register will be cleared. Reset type: SYSRSn
5	CLA1WRITE	R-0/W1S	0h	0: No action. 1: CLA1 Write Access Violation Flag in NMAVFLG register will be cleared. Reset type: SYSRSn
4	CLA1READ	R-0/W1S	0h	0: No action. 1: CLA1 Read Access Violation Flag in NMAVFLG register will be cleared. Reset type: SYSRSn
3	DMAWRITE	R-0/W1S	0h	0: No action. 1: DMA Write Access Violation Flag in NMAVFLG register will be cleared. Reset type: SYSRSn
2	CPUFETCH	R-0/W1S	0h	0: No action. 1: CPU Fetch Access Violation Flag in NMAVFLG register will be cleared. Reset type: SYSRSn
1	CPUWRITE	R-0/W1S	0h	0: No action. 1: CPU Write Access Violation Flag in NMAVFLG register will be cleared. Reset type: SYSRSn



**Table 3-23. NMAVCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	CPUREAD	R-0/W1S	0h	0: No action. 1: CPU Read Access Violation Flag in NMAVFLG register will be cleared. Reset type: SYSRSn

### 3.14.2.4 NMAVINTEN Register (Offset = 6h) [reset = 0h]

NMAVINTEN is shown in [Figure 3-26](#) and described in [Table 3-24](#).

Return to the [Summary Table](#).

Non-Master Access Violation Interrupt Enable Register

**Figure 3-26. NMAVINTEN 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						RESERVED	RESERVED
R-0h							
7	6	5	4	3	2	1	0
RESERVED	CLA1FETCH	CLA1WRITE	CLA1READ	RESERVED	CPUFETCH	CPUWRITE	CPUREAD
	R/W-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h

**Table 3-24. NMAVINTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-10	RESERVED	R	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	CLA1FETCH	R/W	0h	0: CLA1 Non Master Fetch Access Violation Interrupt is disabled. 1: CLA1 Non Master Fetch Access Violation Interrupt is enabled. Reset type: SYSRSn
5	CLA1WRITE	R/W	0h	0: CLA1 Non Master Write Access Violation Interrupt is disabled. 1: CLA1 Non Master Write Access Violation Interrupt is enabled. Reset type: SYSRSn
4	CLA1READ	R/W	0h	0: CLA1 Non Master Read Access Violation Interrupt is disabled. 1: CLA1 Non Master Read Access Violation Interrupt is enabled. Reset type: SYSRSn
3	RESERVED	R/W	0h	Reserved
2	CPUFETCH	R/W	0h	0: CPU Non Master Fetch Access Violation Interrupt is disabled. 1: CPU Non Master Fetch Access Violation Interrupt is enabled. Reset type: SYSRSn
1	CPUWRITE	R/W	0h	0: CPU Non Master Write Access Violation Interrupt is disabled. 1: CPU Non Master Write Access Violation Interrupt is enabled. Reset type: SYSRSn
0	CPUREAD	R/W	0h	0: CPU Non Master Read Access Violation Interrupt is disabled. 1: CPU Non Master Read Access Violation Interrupt is enabled. Reset type: SYSRSn

### 3.14.2.5 NMCPURDAVADDR Register (Offset = 8h) [reset = 0h]

NMCPURDAVADDR is shown in [Figure 3-27](#) and described in [Table 3-25](#).

Return to the [Summary Table](#).

Non-Master CPU Read Access Violation Address

**Figure 3-27. NMCPURDAVADDR 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
NMCPURDAVADDR																															
R-0h																															

**Table 3-25. NMCPURDAVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	NMCPURDAVADDR	R	0h	This register captures the address location for which non master CPU read access violation occurred. Reset type: SYSRSn

### 3.14.2.6 NMCPUWRAVADDR Register (Offset = Ah) [reset = 0h]

NMCPUWRAVADDR is shown in [Figure 3-28](#) and described in [Table 3-26](#).

Return to the [Summary Table](#).

Non-Master CPU Write Access Violation Address

**Figure 3-28. NMCPUWRAVADDR 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
NMCPUWRAVADDR																															
R-0h																															

**Table 3-26. NMCPUWRAVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	NMCPUWRAVADDR	R	0h	This register captures the address location for which non master CPU write access violation occurred. Reset type: SYSRSn

### 3.14.2.7 NMCPUFAVADDR Register (Offset = Ch) [reset = 0h]

NMCPUFAVADDR is shown in [Figure 3-29](#) and described in [Table 3-27](#).

Return to the [Summary Table](#).

Non-Master CPU Fetch Access Violation Address

**Figure 3-29. NMCPUFAVADDR 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
NMCPUFAVADDR																															
R-0h																															

**Table 3-27. NMCPUFAVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	NMCPUFAVADDR	R	0h	This register captures the address location for which non master CPU fetch access violation occurred. Reset type: SYSRSn

### 3.14.2.8 NMDMAWRAVADDR Register (Offset = Eh) [reset = 0h]

NMDMAWRAVADDR is shown in [Figure 3-30](#) and described in [Table 3-28](#).

Return to the [Summary Table](#).

Non-Master DMA Write Access Violation Address

**Figure 3-30. NMDMAWRAVADDR 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																															

**Table 3-28. NMDMAWRAVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	NMDMAWRAVADDR	R	0h	This register captures the address location for which non master DMA write access violation occurred. Reset type: SYSRSn

### 3.14.2.9 NMCLA1RDAVADDR Register (Offset = 10h) [reset = 0h]

NMCLA1RDAVADDR is shown in [Figure 3-31](#) and described in [Table 3-29](#).

Return to the [Summary Table](#).

Non-Master CLA1 Read Access Violation Address

**Figure 3-31. NMCLA1RDAVADDR 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
NMCLA1RDAVADDR																															
R-0h																															

**Table 3-29. NMCLA1RDAVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	NMCLA1RDAVADDR	R	0h	This register captures the address location for which non master CLA1 read access violation occurred. Reset type: SYSRSn



### 3.14.2.10 NMCLA1WRAVADDR Register (Offset = 12h) [reset = 0h]

NMCLA1WRAVADDR is shown in [Figure 3-32](#) and described in [Table 3-30](#).

Return to the [Summary Table](#).

Non-Master CLA1 Write Access Violation Address

**Figure 3-32. NMCLA1WRAVADDR 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
NMCLA1WRAVADDR																															
R-0h																															

**Table 3-30. NMCLA1WRAVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	NMCLA1WRAVADDR	R	0h	This register captures the address location for which non master CLA1 write access violation occurred. Reset type: SYSRSn

### 3.14.2.11 NMCLA1FAVADDR Register (Offset = 14h) [reset = 0h]

NMCLA1FAVADDR is shown in [Figure 3-33](#) and described in [Table 3-31](#).

Return to the [Summary Table](#).

Non-Master CLA1 Fetch Access Violation Address

**Figure 3-33. NMCLA1FAVADDR 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
NMCLA1FAVADDR																															
R-0h																															

**Table 3-31. NMCLA1FAVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	NMCLA1FAVADDR	R	0h	This register captures the address location for which non master CLA1 fetch access violation occurred. Reset type: SYSRSn

**3.14.2.12 MAVFLG Register (Offset = 20h) [reset = 0h]**

MAVFLG is shown in [Figure 3-34](#) and described in [Table 3-32](#).

Return to the [Summary Table](#).

Master Access Violation Flag Register

**Figure 3-34. MAVFLG 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					DMAWRITE	CPUWRITE	CPUFETCH
R-0h					R-0h	R-0h	R-0h

**Table 3-32. MAVFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-3	RESERVED	R	0h	Reserved
2	DMAWRITE	R	0h	Master DMA Write Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn
1	CPUWRITE	R	0h	Master CPU Write Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn
0	CPUFETCH	R	0h	Master CPU Fetch Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn

**3.14.2.13 MAVSET Register (Offset = 22h) [reset = 0h]**

MAVSET is shown in [Figure 3-35](#) and described in [Table 3-33](#).

Return to the [Summary Table](#).

Master Access Violation Flag Set Register

**Figure 3-35. MAVSET 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					DMAWRITE	CPUWRITE	CPUFETCH
R-0h					R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-33. MAVSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-3	RESERVED	R	0h	Reserved
2	DMAWRITE	R-0/W1S	0h	0: No action. 1: DMA Write Access Violation Flag in MAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn
1	CPUWRITE	R-0/W1S	0h	0: No action. 1: CPU Write Access Violation Flag in MAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn
0	CPUFETCH	R-0/W1S	0h	0: No action. 1: CPU Fetch Access Violation Flag in MAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn

**3.14.2.14 MAVCLR Register (Offset = 24h) [reset = 0h]**

MAVCLR is shown in [Figure 3-36](#) and described in [Table 3-34](#).

Return to the [Summary Table](#).

Master Access Violation Flag Clear Register

**Figure 3-36. MAVCLR 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					DMAWRITE	CPUWRITE	CPUFETCH
R-0h					R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-34. MAVCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-3	RESERVED	R	0h	Reserved
2	DMAWRITE	R-0/W1S	0h	0: No action. 1: DMA Write Access Violation Flag in MAVFLG register will be cleared. Reset type: SYSRSn
1	CPUWRITE	R-0/W1S	0h	0: No action. 1: CPU Write Access Violation Flag in MAVFLG register will be cleared . Reset type: SYSRSn
0	CPUFETCH	R-0/W1S	0h	0: No action. 1: CPU Fetch Access Violation Flag in MAVFLG register will be cleared. Reset type: SYSRSn

**3.14.2.15 MAVINTEN Register (Offset = 26h) [reset = 0h]**

MAVINTEN is shown in [Figure 3-37](#) and described in [Table 3-35](#).

Return to the [Summary Table](#).

Master Access Violation Interrupt Enable Register

**Figure 3-37. MAVINTEN 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					DMAWRITE	CPUWRITE	CPUFETCH
R-0h					R/W-0h	R/W-0h	R/W-0h

**Table 3-35. MAVINTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-3	RESERVED	R	0h	Reserved
2	DMAWRITE	R/W	0h	0: DMA Write Access Violation Interrupt is disabled. 1: DMA Write Access Violation Interrupt is enabled. Reset type: SYSRSn
1	CPUWRITE	R/W	0h	0: CPU Write Access Violation Interrupt is disabled. 1: CPU Write Access Violation Interrupt is enabled. Reset type: SYSRSn
0	CPUFETCH	R/W	0h	0: CPU Fetch Access Violation Interrupt is disabled. 1: CPU Fetch Access Violation Interrupt is enabled. Reset type: SYSRSn

### 3.14.2.16 MCPUFAVADDR Register (Offset = 28h) [reset = 0h]

MCPUFAVADDR is shown in [Figure 3-38](#) and described in [Table 3-36](#).

Return to the [Summary Table](#).

Master CPU Fetch Access Violation Address

**Figure 3-38. MCPUFAVADDR 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
MCPUFAVADDR																															
R-0h																															

**Table 3-36. MCPUFAVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	MCPUFAVADDR	R	0h	This register captures the address location for which master CPU fetch access violation occurred. Reset type: SYSRSn



### 3.14.2.17 MCPUWRAVADDR Register (Offset = 2Ah) [reset = 0h]

MCPUWRAVADDR is shown in [Figure 3-39](#) and described in [Table 3-37](#).

Return to the [Summary Table](#).

Master CPU Write Access Violation Address

**Figure 3-39. MCPUWRAVADDR 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
MCPUWRAVADDR																															
R-0h																															

**Table 3-37. MCPUWRAVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	MCPUWRAVADDR	R	0h	This register captures the address location for which master CPU write access violation occurred. Reset type: SYSRSn

**3.14.2.18 MDMAWRVADDR Register (Offset = 2Ch) [reset = 0h]**

MDMAWRVADDR is shown in [Figure 3-40](#) and described in [Table 3-38](#).

Return to the [Summary Table](#).

Master DMA Write Access Violation Address

**Figure 3-40. MDMAWRVADDR 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
MDMAWRVADDR																															
R-0h																															

**Table 3-38. MDMAWRVADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	MDMAWRVADDR	R	0h	This register captures the address location for which master DMA write access violation occurred. Reset type: SYSRSn

### 3.14.3 CLK\_CFG\_REGS Registers

Table 3-39 lists the CLK\_CFG\_REGS registers. All register offset addresses not listed in Table 3-39 should be considered as reserved locations and the register contents should not be modified.

**Table 3-39. CLK\_CFG\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
2h	CLKCFGLOCK1	Lock bit for CLKCFG registers	EALLOW	<a href="#">Go</a>
8h	CLKSRCCTL1	Clock Source Control register-1	EALLOW	<a href="#">Go</a>
Ah	CLKSRCCTL2	Clock Source Control register-2	EALLOW	<a href="#">Go</a>
Ch	CLKSRCCTL3	Clock Source Control register-3	EALLOW	<a href="#">Go</a>
Eh	SYSPLLCTL1	SYSPLL Control register-1	EALLOW	<a href="#">Go</a>
14h	SYSPLLMULT	SYSPLL Multiplier register	EALLOW	<a href="#">Go</a>
16h	SYSPLLSTS	SYSPLL Status register		<a href="#">Go</a>
22h	SYSCLKDIVSEL	System Clock Divider Select register	EALLOW	<a href="#">Go</a>
28h	XCLKOUTDIVSEL	XCLKOUT Divider Select register	EALLOW	<a href="#">Go</a>
2Ch	LOSPCP	Low Speed Clock Source Prescaler	EALLOW	<a href="#">Go</a>
2Eh	MCDCCR	Missing Clock Detect Control Register	EALLOW	<a href="#">Go</a>
30h	X1CNT	10-bit Counter on X1 Clock		<a href="#">Go</a>
32h	XTALCR	XTAL Control Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-40 shows the codes that are used for access types in this section.

**Table 3-40. CLK\_CFG\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
WSonce	WSonce	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.3.1 CLKCFGLOCK1 Register (Offset = 2h) [reset = 0h]

CLKCFGLOCK1 is shown in [Figure 3-41](#) and described in [Table 3-41](#).

Return to the [Summary Table](#).

Lock bit for CLKCFG registers

Notes:

[1] Any bit in this register, once set can only be cleared through a CPU1.SYSRSn. Write of 0 to any bit of this register has no effect

[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed

**Figure 3-41. CLKCFGLOCK1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							XTALCR
R-0-0h							R/WOnce-0h
15	14	13	12	11	10	9	8
LOSPCP	RESERVED	RESERVED	RESERVED	SYSCLKDIVSEL	RESERVED	RESERVED	RESERVED
R/WOnce-0h	R-0-0h			R/WOnce-0h		R-0-0h	R-0-0h
7	6	5	4	3	2	1	0
RESERVED	SYSPLLMULT	RESERVED	RESERVED	SYSPLLCTL1	CLKSRCCTL3	CLKSRCCTL2	CLKSRCCTL1
	R/WOnce-0h			R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 3-41. CLKCFGLOCK1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R-0	0h	Reserved
16	XTALCR	R/WOnce	0h	Lock bit for XTALCR register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
15	LOSPCP	R/WOnce	0h	Lock bit for LOSPCP register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
14	RESERVED	R-0	0h	Reserved
13	RESERVED	R/WOnce	0h	Reserved
12	RESERVED	R/WOnce	0h	Reserved
11	SYSCLKDIVSEL	R/WOnce	0h	Lock bit for SYSCLKDIVSEL register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
10	RESERVED	R/WOnce	0h	Reserved
9	RESERVED	R-0	0h	Reserved
8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/WOnce	0h	Reserved

**Table 3-41. CLKCFGLOCK1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	SYSPLLMULT	R/WOnce	0h	Lock bit for SYSPLLMULT register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
5	RESERVED	R/WOnce	0h	Reserved
4	RESERVED	R/WOnce	0h	Reserved
3	SYSPLLCTL1	R/WOnce	0h	Lock bit for SYSPLLCTL1 register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
2	CLKSRCCTL3	R/WOnce	0h	Lock bit for CLKSRCCTL3 register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
1	CLKSRCCTL2	R/WOnce	0h	Lock bit for CLKSRCCTL2 register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
0	CLKSRCCTL1	R/WOnce	0h	Lock bit for CLKSRCCTL1 register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn

### 3.14.3.2 CLKSRCCTL1 Register (Offset = 8h) [reset = 0h]

CLKSRCCTL1 is shown in [Figure 3-42](#) and described in [Table 3-42](#).

Return to the [Summary Table](#).

Clock Source Control register-1

**Figure 3-42. CLKSRCCTL1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		WDHALTI	RESERVED	INTOSC2OFF	RESERVED	OSCCLKSRCSEL	
R-0-0h		R/W-0h		R/W-0h	R-0-0h	R/W-0h	

**Table 3-42. CLKSRCCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5	WDHALTI	R/W	0h	Watchdog HALT Mode Ignore Bit: This bit determines if WD is functional in the HALT mode or not.  0 = WD is not functional in the HALT mode. Clock to WD is gated when system enters HALT mode. Additionally, INTOSC1 and INTOSC2 are powered-down when system enters HALT mode 1 = WD is functional in the HALT mode. Clock to WD is not gated and INTOSC1/2 are not powered-down when system enters HALT mode Reset type: XRSn
4	RESERVED	R/W	0h	Reserved
3	INTOSC2OFF	R/W	0h	Internal Oscillator 2 Off Bit: This bit turns oscillator 2 off: 0 = Internal Oscillator 2 On (default on reset) 1 = Internal Oscillator 2 Off  This bit could be used by the user to turn off the internal oscillator 2 if it is not used.  NOTE: Ensure no resources are using a clock source prior to disabling it. For example OSCCLKSRCSEL (SYSPLL), AUXOSCCLKSRCSEL (AUXPLL), TMR2CLKSRCSEL (CPUTIMER2) and XCLOCKOUT (XCLKOUT). Reset type: XRSn
2	RESERVED	R-0	0h	Reserved

**Table 3-42. CLKSRCCTL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	OSCCLKSRCSEL	R/W	0h	<p>Oscillator Clock Source Select Bit: This bit selects the source for OSCCLK.</p> <p>00 = INTOSC2 (default on reset)</p> <p>01 = External Oscillator (XTAL)</p> <p>10 = INTOSC1</p> <p>11 = reserved (default to INTOSC1)</p> <p>At power-up or after an XRSn, INTOSC2 is selected by default. Whenever the user changes the clock source using these bits, the SYSPLLMULT register will be forced to zero and the PLL will be bypassed and powered down. This prevents potential PLL overshoot. The user will then have to write to the SYSPLLMULT register to configure the appropriate multiplier.</p> <p>The user must wait 10 OSCCLK cycles before writing to SYSPLLMULT or disabling the previous clock source to allow the change to complete..</p> <p>Notes:</p> <p>[1] Reserved selection defaults to 00 configuration</p> <p>[2] INTOSC1 is recommended to be used only after missing clock detection. If user wants to re-lock the PLL with INTOSC1 (the back-up clock source) after missing clock is detected, he can do a MCLKCLR and lock the PLL.</p> <p>Reset type: XRSn</p>

### 3.14.3.3 CLKSRCCTL2 Register (Offset = Ah) [reset = 0h]

CLKSRCCTL2 is shown in [Figure 3-43](#) and described in [Table 3-43](#).

Return to the [Summary Table](#).

Clock Source Control register-2

**Figure 3-43. CLKSRCCTL2 Register**

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

**Table 3-43. CLKSRCCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R-0	0h	Reserved
9-8	RESERVED	R/W	0h	Reserved
7-6	RESERVED	R/W	0h	Reserved
5-4	CANBBCLKSEL	R/W	0h	CANB Bit-Clock Source Select Bit: 00 = PERx.SYSCLK (default on reset) 01 = External Oscillator (XTAL) 10 = Reserved 11 = Reserved  Missing clock detect circuit doesnt have any impact on these bits. Reset type: XRSn
3-2	CANABCLKSEL	R/W	0h	CANA Bit-Clock Source Select Bit: 00 = PERx.SYSCLK (default on reset) 01 = External Oscillator (XTAL) 10 = Reserved 11 = Reserved  Missing clock detect circuit doesnt have any impact on these bits. Reset type: XRSn
1-0	RESERVED	R/W	0h	Reserved



**3.14.3.4 CLKSRCCTL3 Register (Offset = Ch) [reset = 0h]**

CLKSRCCTL3 is shown in [Figure 3-44](#) and described in [Table 3-44](#).

Return to the [Summary Table](#).

Clock Source Control register-3

**Figure 3-44. CLKSRCCTL3 Register**

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

**Table 3-44. CLKSRCCTL3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2-0	XCLKOUTSEL	R/W	0h	XCLKOUT Source Select Bit: This bit selects the source for XCLKOUT: 000 = PLLSYSCLK (default on reset) 001 = PLLRAWCLK 010 = SYSCLK 011 = Reserved 100 = Reserved 101 = INTOSC1 110 = INTOSC2 111 = XTAL OSC o/p clock Reset type: SYSRSn

### 3.14.3.5 SYSPLLCTL1 Register (Offset = Eh) [reset = 0h]

SYSPLLCTL1 is shown in [Figure 3-45](#) and described in [Table 3-45](#).

Return to the [Summary Table](#).

SYSPLL Control register-1

**Figure 3-45. SYSPLLCTL1 Register**

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

**Table 3-45. SYSPLLCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	PLLCLKEN	R/W	0h	SYSPLL bypassed or included in the PLLSYSCLK path: This bit decides if the SYSPLL is bypassed when PLLSYSCLK is generated  1 = PLLSYSCLK is fed from the SYSPLL clock output. Users need to make sure that the PLL is locked before enabling this clock to the system.  0 = SYSPLL is bypassed. Clock to system is direct feed from OSCCLK  Reset type: XRSn
0	PLLEN	R/W	0h	SYSPLL enabled or disabled: This bit decides if the SYSPLL is enabled or not  1 = SYSPLL is enabled  0 = SYSPLL is powered off. Clock to system is direct feed from OSCCLK  Reset type: XRSn

### 3.14.3.6 SYSPLLMULT Register (Offset = 14h) [reset = 0h]

SYSPLLMULT is shown in [Figure 3-46](#) and described in [Table 3-46](#).

Return to the [Summary Table](#).

SYSPLL Multiplier register

NOTE: FMULT and IMULT fields must be written at the same time for correct PLL operation.

**Figure 3-46. SYSPLLMULT Register**

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

**Table 3-46. SYSPLLMULT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R-0	0h	Reserved
29-24	RESERVED	R/W	0h	Reserved
23-19	RESERVED	R-0	0h	Reserved
18-16	ODIV	R/W	0h	SYSPLL Output Clock Divider PLL Output Divider = ODIV + 1 Reset type: XRSn
15-10	RESERVED	R-0	0h	Reserved
9-8	FMULT	R/W	0h	SYSPLL Fractional Multiplier: 00 Fractional Multiplier = 0 01 Fractional Multiplier = 0.25 10 Fractional Multiplier = 0.5 11 Fractional Multiplier = 0.75 Reset type: XRSn
7	RESERVED	R-0	0h	Reserved
6-0	IMULT	R/W	0h	SYSPLL Integer Multiplier: For 0000000 Fout = Fref (PLLBYPASS) Integer Multiplier = 1 0000001 Integer Multiplier = 1 0000010 Integer Multiplier = 2 0000011 Integer Multiplier = 3 ..... 1111111 Integer Multiplier = 127 Reset type: XRSn

### 3.14.3.7 SYSPLLSTS Register (Offset = 16h) [reset = 0h]

SYSPLLSTS is shown in [Figure 3-47](#) and described in [Table 3-47](#).

Return to the [Summary Table](#).

SYSPLL Status register

**Figure 3-47. SYSPLLSTS Register**

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

**Table 3-47. SYSPLLSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	SLIPS	R	0h	SYSPLL Slip Status Bit: This bit indicates whether the SYSPLL is out of lock range  0 = SYSPLL is not out of lock 1 = SYSPLL is out of lock Reset type: XRSn
0	LOCKS	R	0h	SYSPLL Lock Status Bit: This bit indicates whether the SYSPLL is locked or not  0 = SYSPLL is not yet locked 1 = SYSPLL is locked Reset type: XRSn

**3.14.3.8 SYCLKDIVSEL Register (Offset = 22h) [reset = 2h]**

SYCLKDIVSEL is shown in [Figure 3-48](#) and described in [Table 3-48](#).

Return to the [Summary Table](#).

System Clock Divider Select register

**Figure 3-48. SYCLKDIVSEL Register**

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

**Table 3-48. SYCLKDIVSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-0	PLLSYSCLKDIV	R/W	2h	PLLSYSCLK Divide Select: This bit selects the divider setting for the PLLSYSCLK.  000000 = /1 000001 = /2 000010 = /4 (default on reset) 000011 = /6 000100 = /8 ..... 111111 = /126 Reset type: XRSn

### 3.14.3.9 XCLKOUTDIVSEL Register (Offset = 28h) [reset = 3h]

XCLKOUTDIVSEL is shown in [Figure 3-49](#) and described in [Table 3-49](#).

Return to the [Summary Table](#).

XCLKOUT Divider Select register

**Figure 3-49. XCLKOUTDIVSEL Register**

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

**Table 3-49. XCLKOUTDIVSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1-0	XCLKOUTDIV	R/W	3h	XCLKOUT Divide Select: This bit selects the divider setting for the XCLKOUT.  00 = /1 01 = /2 10 = /4 11 = /8 (default on reset) Reset type: SYSRSn

### 3.14.3.10 LOSPCP Register (Offset = 2Ch) [reset = 2h]

LOSPCP is shown in [Figure 3-50](#) and described in [Table 3-50](#).

Return to the [Summary Table](#).

Low Speed Clock Source Prescaler

**Figure 3-50. LOSPCP Register**

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

**Table 3-50. LOSPCP Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2-0	LSPCLKDIV	R/W	2h	These bits configure the low-speed peripheral clock (LSPCLK) rate 000,LSPCLK = / 1 001,LSPCLK = / 2 010,LSPCLK = / 4 (default on reset) 011,LSPCLK = / 6 100,LSPCLK = / 8 101,LSPCLK = / 10 110,LSPCLK = / 12 111,LSPCLK = / 14  Note: [1] This clock is used as strobe for the SCI and SPI modules. Reset type: SYSRSn

**3.14.3.11 MCDCR Register (Offset = 2Eh) [reset = 0h]**

MCDCR is shown in [Figure 3-51](#) and described in [Table 3-51](#).

Return to the [Summary Table](#).

Missing Clock Detect Control Register

**Figure 3-51. MCDCR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				OSCOFF	MCLKOFF	MCLKCLR	MCLKSTS
R-0-0h				R/W-0h	R/W-0h	R-0/W1S-0h	R-0h

**Table 3-51. MCDCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	OSCOFF	R/W	0h	Oscillator Clock Disconnect from MCD Bit: 0 = OSCCLK Connected to OSCCLK Counter in MCD module 1 = OSCCLK Disconnected to OSCCLK Counter in MCD module Reset type: XRSn
2	MCLKOFF	R/W	0h	Missing Clock Detect Off Bit: 0 = Missing Clock Detect Circuit Enabled 1 = Missing Clock Detect Circuit Disabled Reset type: XRSn
1	MCLKCLR	R-0/W1S	0h	Missing Clock Clear Bit: Write "1" to this bit to clear MCLKSTS bit and reset the missing clock detect circuit." Reset type: XRSn
0	MCLKSTS	R	0h	Missing Clock Status Bit: 0 = OSCCLK Is OK 1 = OSCCLK Detected Missing, CLOCKFAILn Generated Reset type: XRSn



### 3.14.3.12 X1CNT Register (Offset = 30h) [reset = 0h]

X1CNT is shown in [Figure 3-52](#) and described in [Table 3-52](#).

Return to the [Summary Table](#).

10-bit Counter on X1 Clock

**Figure 3-52. X1CNT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															CLR
R-0-0h															R-0/W1S-0h
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						X1CNT									
R-0-0h						R-0h									

**Table 3-52. X1CNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R-0	0h	Reserved
16	CLR	R-0/W1S	0h	X1 Counter clear: A write of '1' to this bit field clears the X1CNT and makes it count from 0x0 again (provided X1 clock is ticking). Writes of '0' are ignore to this bit field Reset type: N/A
15-10	RESERVED	R-0	0h	Reserved
9-0	X1CNT	R	0h	X1 Counter: - This counter increments on every X1 CLOCKS positive-edge. - Once it reaches the values of 0x3ff, it freezes - Before switching from INTOSC2 to X1, application must check this counter and make sure that it has saturated. This will ensure that the Crystal connected to X1/X2 is oscillating. Reset type: POR

**3.14.3.13 XTALCR Register (Offset = 32h) [reset = 5h]**

XTALCR is shown in [Figure 3-53](#) and described in [Table 3-53](#).

Return to the [Summary Table](#).

XTAL Control Register

**Figure 3-53. XTALCR Register**

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

**Table 3-53. XTALCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2	RESERVED	R/W	1h	Reserved
1	SE	R/W	0h	Configures XTAL oscillator in single-ended or Crystal mode when XTAL oscillator is powered up (i.e. OSCOFF = 0)  0 XTAL oscillator in Crystal mode 1 XTAL oscillator in single-ended mode (through X1) Reset type: XRSn
0	OSCOFF	R/W	1h	This bit if '1', powers-down the XTAL oscillator macro and hence doesn't let X2 to be driven by the XTAL oscillator. If a crystal is connected to X1/X2, user needs to first clear this bit, wait for the oscillator to power up (using X1CNT) and then only switch the clock source to X1/X2  Reset type: XRSn

### 3.14.4 CPU\_SYS\_REGS Registers

Table 3-54 lists the CPU\_SYS\_REGS registers. All register offset addresses not listed in Table 3-54 should be considered as reserved locations and the register contents should not be modified.

**Table 3-54. CPU\_SYS\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	CPUSYSLOCK1	Lock bit for CPUSYS registers	EALLOW	<a href="#">Go</a>
Ah	PIEVERRADDR	PIE Vector Fetch Error Address register	EALLOW	<a href="#">Go</a>
22h	PCLKCR0	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
26h	PCLKCR2	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
28h	PCLKCR3	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
2Ah	PCLKCR4	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
2Eh	PCLKCR6	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
30h	PCLKCR7	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
32h	PCLKCR8	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
34h	PCLKCR9	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
36h	PCLKCR10	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
3Ch	PCLKCR13	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
3Eh	PCLKCR14	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
40h	PCLKCR15	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
42h	PCLKCR16	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
44h	PCLKCR17	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
46h	PCLKCR18	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
48h	PCLKCR19	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
4Ah	PCLKCR20	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
4Ch	PCLKCR21	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
76h	LPMCR	LPM Control Register	EALLOW	<a href="#">Go</a>
78h	GPIOLPMSEL0	GPIO LPM Wakeup select registers	EALLOW	<a href="#">Go</a>
7Ah	GPIOLPMSEL1	GPIO LPM Wakeup select registers	EALLOW	<a href="#">Go</a>
7Ch	TMR2CLKCTL	Timer2 Clock Measurement functionality control register	EALLOW	<a href="#">Go</a>
7Eh	RESCCLR	Reset Cause Clear Register		<a href="#">Go</a>
80h	RESC	Reset Cause register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-55 shows the codes that are used for access types in this section.

**Table 3-55. CPU\_SYS\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
WOnce	WOnce	Write Set once
<b>Reset or Default Value</b>		

**Table 3-55. CPU\_SYS\_REGS Access Type Codes (continued)**

Access Type	Code	Description
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.4.1 CPUSYSLOCK1 Register (Offset = 0h) [reset = 0h]

CPUSYSLOCK1 is shown in [Figure 3-54](#) and described in [Table 3-56](#).

Return to the [Summary Table](#).

Lock bit for CPUSYS registers

Notes:

[1] Any bit in this register, once set can only be cleared through a CPU1.SYSRSn. Write of 0 to any bit of this register has no effect

[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed

**Figure 3-54. CPUSYSLOCK1 Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	PCLKCR21	PCLKCR20	PCLKCR19	PCLKCR18	PCLKCR17
			R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h
23	22	21	20	19	18	17	16
GPIOLPMSEL1	GPIOLPMSEL0	LPMCR	RESERVED	PCLKCR16	PCLKCR15	PCLKCR14	PCLKCR13
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h		R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	PCLKCR10	PCLKCR9	PCLKCR8	PCLKCR7	PCLKCR6	RESERVED
		R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	
7	6	5	4	3	2	1	0
PCLKCR4	PCLKCR3	PCLKCR2	RESERVED	PCLKCR0	PIEVERRADD R	RESERVED	RESERVED
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h		R/WOnce-0h	R/WOnce-0h		

**Table 3-56. CPUSYSLOCK1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/WOnce	0h	Reserved
30	RESERVED	R/WOnce	0h	Reserved
29	RESERVED	R/WOnce	0h	Reserved
28	PCLKCR21	R/WOnce	0h	Lock bit for PCLKCR21 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
27	PCLKCR20	R/WOnce	0h	Lock bit for PCLKCR20 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
26	PCLKCR19	R/WOnce	0h	Lock bit for PCLKCR19 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
25	PCLKCR18	R/WOnce	0h	Lock bit for PCLKCR18 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn

**Table 3-56. CPUSYSLOCK1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
24	PCLKCR17	R/WOnce	0h	Lock bit for PCLKCR17 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
23	GPIOLPMSEL1	R/WOnce	0h	Lock bit for GPIOLPMSEL1 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
22	GPIOLPMSEL0	R/WOnce	0h	Lock bit for GPIOLPMSEL0 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
21	LPMCR	R/WOnce	0h	Lock bit for LPMCR Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
20	RESERVED	R/WOnce	0h	Reserved
19	PCLKCR16	R/WOnce	0h	Lock bit for PCLKCR16 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
18	PCLKCR15	R/WOnce	0h	Lock bit for PCLKCR15 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
17	PCLKCR14	R/WOnce	0h	Lock bit for PCLKCR14 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
16	PCLKCR13	R/WOnce	0h	Lock bit for PCLKCR13 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
15	RESERVED	R/WOnce	0h	Reserved
14	RESERVED	R/WOnce	0h	Reserved
13	PCLKCR10	R/WOnce	0h	Lock bit for PCLKCR10 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn

**Table 3-56. CPUSYSLOCK1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	PCLKCR9	R/WOnce	0h	Lock bit for PCLKCR9 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
11	PCLKCR8	R/WOnce	0h	Lock bit for PCLKCR8 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
10	PCLKCR7	R/WOnce	0h	Lock bit for PCLKCR7 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
9	PCLKCR6	R/WOnce	0h	Lock bit for PCLKCR6 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
8	RESERVED	R/WOnce	0h	Reserved
7	PCLKCR4	R/WOnce	0h	Lock bit for PCLKCR4 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
6	PCLKCR3	R/WOnce	0h	Lock bit for PCLKCR3 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
5	PCLKCR2	R/WOnce	0h	Lock bit for PCLKCR2 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
4	RESERVED	R/WOnce	0h	Reserved
3	PCLKCR0	R/WOnce	0h	Lock bit for PCLKCR0 Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
2	PIEVERRADDR	R/WOnce	0h	Lock bit for PIEVERRADDR Register: 0: Respective register is not locked 1: Respective register is locked.  Reset type: SYSRSn
1	RESERVED	R/WOnce	0h	Reserved
0	RESERVED	R/WOnce	0h	Reserved

### 3.14.4.2 PIEVERRADDR Register (Offset = Ah) [reset = 003FFFFFFh]

PIEVERRADDR is shown in [Figure 3-55](#) and described in [Table 3-57](#).

Return to the [Summary Table](#).

PIE Vector Fetch Error Address register

**Figure 3-55. PIEVERRADDR 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											ADDR																				
R-0-0h											R/W-003FFFFFFh																				

**Table 3-57. PIEVERRADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-22	RESERVED	R-0	0h	Reserved
21-0	ADDR	R/W	003FFFFFFh	This register defines the address of the PIE Vector Fetch Error handler routine. Its the responsibility of user to initialize this register. If this register is not initialized, a default error handler at address 0x3ffbe will get executed. Refer to the Boot ROM section for more details on this register. Reset type: XRSn



### 3.14.4.3 PCLKCR0 Register (Offset = 22h) [reset = 38h]

PCLKCR0 is shown in [Figure 3-56](#) and described in [Table 3-58](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-56. PCLKCR0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED	TBCLKSYNC	RESERVED	HRPWM
R-0-0h				R/W-0h		R-0-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		CPUTIMER2	CPUTIMER1	CPUTIMER0	DMA	RESERVED	CLA1
R-0-0h		R/W-1h	R/W-1h	R/W-1h	R/W-0h		R/W-0h

**Table 3-58. PCLKCR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R-0	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	TBCLKSYNC	R/W	0h	EPWM Time Base Clock sync: When set, it synchronizes all enabled ePWM modules to the time-base clock (TBCLK). Reset type: SYSRSn
17	RESERVED	R-0	0h	Reserved
16	HRPWM	R/W	0h	HRPWM Clock Enable Bit: When set, this enables the clock to the HRPWM module 1: HRPWM clock is enabled 0: HRPWM clock is disabled Reset type: SYSRSn
15-6	RESERVED	R-0	0h	Reserved
5	CPUTIMER2	R/W	1h	CPUTIMER2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
4	CPUTIMER1	R/W	1h	CPUTIMER1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
3	CPUTIMER0	R/W	1h	CPUTIMER0 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**Table 3-58. PCLKCR0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	DMA	R/W	0h	DMA Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on  Reset type: SYSRSn
1	RESERVED	R/W	0h	Reserved
0	CLA1	R/W	0h	CLA1 Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on  Reset type: SYSRSn

### 3.14.4.4 PCLKCR2 Register (Offset = 26h) [reset = 0h]

PCLKCR2 is shown in [Figure 3-57](#) and described in [Table 3-59](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-57. PCLKCR2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
7	6	5	4	3	2	1	0
EPWM8	EPWM7	EPWM6	EPWM5	EPWM4	EPWM3	EPWM2	EPWM1
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 3-59. PCLKCR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	EPWM8	R/W	0h	EPWM8 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
6	EPWM7	R/W	0h	EPWM7 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
5	EPWM6	R/W	0h	EPWM6 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
4	EPWM5	R/W	0h	EPWM5 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**Table 3-59. PCLKCR2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	EPWM4	R/W	0h	EPWM4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
2	EPWM3	R/W	0h	EPWM3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	EPWM2	R/W	0h	EPWM2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	EPWM1	R/W	0h	EPWM1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.14.4.5 PCLKCR3 Register (Offset = 28h) [reset = 0h]

PCLKCR3 is shown in [Figure 3-58](#) and described in [Table 3-60](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-58. PCLKCR3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	ECAP7	ECAP6	ECAP5	ECAP4	ECAP3	ECAP2	ECAP1
	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-60. PCLKCR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	ECAP7	R/W	0h	ECAP7 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
5	ECAP6	R/W	0h	ECAP6 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
4	ECAP5	R/W	0h	ECAP5 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
3	ECAP4	R/W	0h	ECAP4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
2	ECAP3	R/W	0h	ECAP3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	ECAP2	R/W	0h	ECAP2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**Table 3-60. PCLKCR3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	ECAP1	R/W	0h	ECAP1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.14.4.6 PCLKCR4 Register (Offset = 2Ah) [reset = 0h]

PCLKCR4 is shown in [Figure 3-59](#) and described in [Table 3-61](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-59. PCLKCR4 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED	RESERVED	EQEP2	EQEP1
R-0-0h						R/W-0h	R/W-0h

**Table 3-61. PCLKCR4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	EQEP2	R/W	0h	EQEP2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	EQEP1	R/W	0h	EQEP1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.14.4.7 PCLKCR6 Register (Offset = 2Eh) [reset = 0h]

PCLKCR6 is shown in [Figure 3-60](#) and described in [Table 3-62](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-60. PCLKCR6 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								RESE RVED	RESE RVED	RESE RVED	RESE RVED	RESE RVED	RESE RVED	RESE RVED	SD1
R-0-0h															R/W- 0h

**Table 3-62. PCLKCR6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	SD1	R/W	0h	SD1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn



**3.14.4.8 PCLKCR7 Register (Offset = 30h) [reset = 0h]**

PCLKCR7 is shown in [Figure 3-61](#) and described in [Table 3-63](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-61. PCLKCR7 Register**

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

**Table 3-63. PCLKCR7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	SCI_B	R/W	0h	SCI_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	SCI_A	R/W	0h	SCI_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.9 PCLKCR8 Register (Offset = 32h) [reset = 0h]**

PCLKCR8 is shown in [Figure 3-62](#) and described in [Table 3-64](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-62. PCLKCR8 Register**

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

**Table 3-64. PCLKCR8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-18	RESERVED	R-0	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	SPI_B	R/W	0h	SPI_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	SPI_A	R/W	0h	SPI_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.10 PCLKCR9 Register (Offset = 34h) [reset = 0h]**

PCLKCR9 is shown in [Figure 3-63](#) and described in [Table 3-65](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-63. PCLKCR9 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						RESERVED	I2C_A
R-0-0h							R/W-0h

**Table 3-65. PCLKCR9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	I2C_A	R/W	0h	I2C_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.11 PCLKCR10 Register (Offset = 36h) [reset = 0h]**

PCLKCR10 is shown in [Figure 3-64](#) and described in [Table 3-66](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-64. PCLKCR10 Register**

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

**Table 3-66. PCLKCR10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	CAN_B	R/W	0h	CAN_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	CAN_A	R/W	0h	CAN_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.12 PCLKCR13 Register (Offset = 3Ch) [reset = 0h]**

PCLKCR13 is shown in [Figure 3-65](#) and described in [Table 3-67](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-65. PCLKCR13 Register**

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

**Table 3-67. PCLKCR13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	ADC_C	R/W	0h	ADC_C Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	ADC_B	R/W	0h	ADC_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	ADC_A	R/W	0h	ADC_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.13 PCLKCR14 Register (Offset = 3Eh) [reset = 0h]**

PCLKCR14 is shown in [Figure 3-66](#) and described in [Table 3-68](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-66. PCLKCR14 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	CMPSS7	CMPSS6	CMPSS5	CMPSS4	CMPSS3	CMPSS2	CMPSS1
	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-68. PCLKCR14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	CMPSS7	R/W	0h	CMPSS7 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
5	CMPSS6	R/W	0h	CMPSS6 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
4	CMPSS5	R/W	0h	CMPSS5 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
3	CMPSS4	R/W	0h	CMPSS4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
2	CMPSS3	R/W	0h	CMPSS3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	CMPSS2	R/W	0h	CMPSS2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**Table 3-68. PCLKCR14 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	CMPSS1	R/W	0h	CMPSS1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.14 PCLKCR15 Register (Offset = 40h) [reset = 0h]**

PCLKCR15 is shown in [Figure 3-67](#) and described in [Table 3-69](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-67. PCLKCR15 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	PGA7	PGA6	PGA5	PGA4	PGA3	PGA2	PGA1
	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-69. PCLKCR15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	PGA7	R/W	0h	PGA7 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
5	PGA6	R/W	0h	PGA6 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
4	PGA5	R/W	0h	PGA5 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
3	PGA4	R/W	0h	PGA4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
2	PGA3	R/W	0h	PGA3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	PGA2	R/W	0h	PGA2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn



**Table 3-69. PCLKCR15 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	PGA1	R/W	0h	PGA1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.15 PCLKCR16 Register (Offset = 42h) [reset = 0h]**

PCLKCR16 is shown in [Figure 3-68](#) and described in [Table 3-70](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-68. PCLKCR16 Register**

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

**Table 3-70. PCLKCR16 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R-0	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	DAC_B	R/W	0h	Buffered_DAC_B Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
16	DAC_A	R/W	0h	Buffered_DAC_A Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
15-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

**3.14.4.16 PCLKCR17 Register (Offset = 44h) [reset = 0h]**

PCLKCR17 is shown in [Figure 3-69](#) and described in [Table 3-71](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-69. PCLKCR17 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				CLB4	CLB3	CLB2	CLB1
R-0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-71. PCLKCR17 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	CLB4	R/W	0h	CLB4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
2	CLB3	R/W	0h	CLB3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	CLB2	R/W	0h	CLB2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	CLB1	R/W	0h	CLB1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.17 PCLKCR18 Register (Offset = 46h) [reset = 0h]**

PCLKCR18 is shown in [Figure 3-70](#) and described in [Table 3-72](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-70. PCLKCR18 Register**

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

**Table 3-72. PCLKCR18 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	FSIRX_A	R/W	0h	FSIRX_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	FSITX_A	R/W	0h	FSITX_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.18 PCLKCR19 Register (Offset = 48h) [reset = 0h]**

PCLKCR19 is shown in [Figure 3-71](#) and described in [Table 3-73](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-71. PCLKCR19 Register**

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

**Table 3-73. PCLKCR19 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	LIN_A	R/W	0h	LIN_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.19 PCLKCR20 Register (Offset = 4Ah) [reset = 0h]**

PCLKCR20 is shown in [Figure 3-72](#) and described in [Table 3-74](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-72. PCLKCR20 Register**

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

**Table 3-74. PCLKCR20 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	PMBUS_A	R/W	0h	PMBUS_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.20 PCLKCR21 Register (Offset = 4Ch) [reset = 0h]**

PCLKCR21 is shown in [Figure 3-73](#) and described in [Table 3-75](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-73. PCLKCR21 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED							DCC_0
R-0-0h							R/W-0h

**Table 3-75. PCLKCR21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	DCC_0	R/W	0h	DCC Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**3.14.4.21 LPMCR Register (Offset = 76h) [reset = FCh]**

LPMCR is shown in [Figure 3-74](#) and described in [Table 3-76](#).

Return to the [Summary Table](#).

LPM Control Register

**Figure 3-74. LPMCR Register**

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

**Table 3-76. LPMCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W1S	0h	Reserved
30-18	RESERVED	R-0	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14-8	RESERVED	R-0	0h	Reserved
7-2	RESERVED	R/W	3Fh	Reserved
1-0	LPM	R/W	0h	These bits set the low power mode for the device. Takes effect when CPU executes the IDLE instruction (when IDLE instruction is out of EXE Phase of the Pipeline)  00: IDLE Mode 01: Reserved 1x: HALT Mode Reset type: SYSRSn



### 3.14.4.22 GPIO\_LPMSEL0 Register (Offset = 78h) [reset = 0h]

GPIO\_LPMSEL0 is shown in [Figure 3-75](#) and described in [Table 3-77](#).

Return to the [Summary Table](#).

GPIO LPM Wakeup select registers

Connects the selected pin to the LPM circuit. Refer to LPM section of the TRM for the wakeup capabilities of the selected pin.

**Figure 3-75. GPIO\_LPMSEL0 Register**

31		30		29		28		27		26		25		24	
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24	GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16	GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 3-77. GPIO\_LPMSEL0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
30	GPIO30	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
29	GPIO29	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
28	GPIO28	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
27	GPIO27	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
26	GPIO26	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
25	GPIO25	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
24	GPIO24	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

**Table 3-77. GPIO\_LPMSEL0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23	GPIO23	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
22	GPIO22	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
21	GPIO21	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
20	GPIO20	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
19	GPIO19	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
18	GPIO18	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
17	GPIO17	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
16	GPIO16	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
15	GPIO15	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
14	GPIO14	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
13	GPIO13	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
12	GPIO12	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
11	GPIO11	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
10	GPIO10	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
9	GPIO9	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
8	GPIO8	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

**Table 3-77. GPIO\_LPMSEL0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	GPIO7	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
6	GPIO6	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
5	GPIO5	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
4	GPIO4	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
3	GPIO3	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
2	GPIO2	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
1	GPIO1	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
0	GPIO0	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

### 3.14.4.23 GPIO\_LPMSEL1 Register (Offset = 7Ah) [reset = 0h]

GPIO\_LPMSEL1 is shown in [Figure 3-76](#) and described in [Table 3-78](#).

Return to the [Summary Table](#).

GPIO LPM Wakeup select registers

Connects the selected pin to the LPM circuit. Refer to LPM section of the TRM for the wakeup capabilities of the selected pin.

**Figure 3-76. GPIO\_LPMSEL1 Register**

31		30		29		28		27		26		25		24	
GPIO63		GPIO62		GPIO61		GPIO60		GPIO59		GPIO58		GPIO57		GPIO56	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23		22		21		20		19		18		17		16	
GPIO55		GPIO54		GPIO53		GPIO52		GPIO51		GPIO50		GPIO49		GPIO48	
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	
GPIO47		GPIO46		GPIO45		GPIO44		GPIO43		GPIO42		GPIO41		GPIO40	
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	
GPIO39		GPIO38		GPIO37		GPIO36		GPIO35		GPIO34		GPIO33		GPIO32	
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 3-78. GPIO\_LPMSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
30	GPIO62	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
29	GPIO61	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
28	GPIO60	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
27	GPIO59	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
26	GPIO58	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
25	GPIO57	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
24	GPIO56	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

**Table 3-78. GPIO\_LPMSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23	GPIO55	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
22	GPIO54	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
21	GPIO53	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
20	GPIO52	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
19	GPIO51	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
18	GPIO50	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
17	GPIO49	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
16	GPIO48	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
15	GPIO47	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
14	GPIO46	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
13	GPIO45	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
12	GPIO44	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
11	GPIO43	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
10	GPIO42	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
9	GPIO41	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
8	GPIO40	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

**Table 3-78. GPIO\_LPMSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	GPIO39	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
6	GPIO38	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
5	GPIO37	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
4	GPIO36	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
3	GPIO35	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
2	GPIO34	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
1	GPIO33	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
0	GPIO32	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

### 3.14.4.24 TMR2CLKCTL Register (Offset = 7Ch) [reset = 0h]

TMR2CLKCTL is shown in [Figure 3-77](#) and described in [Table 3-79](#).

Return to the [Summary Table](#).

Timer2 Clock Measurement functionality control register

**Figure 3-77. TMR2CLKCTL Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		TMR2CLKPRESCALE			TMR2CLKSRCSEL		
R-0-0h		R/W-0h			R/W-0h		

**Table 3-79. TMR2CLKCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-3	TMR2CLKPRESCALE	R/W	0h	<p>CPU Timer 2 Clock Pre-Scale Value: These bits select the pre-scale value for the selected clock source for CPU Timer 2:</p> <p>0,0,0,/1 (default on reset)</p> <p>0,0,1,/2,</p> <p>0,1,0,/4</p> <p>0,1,1,/8</p> <p>1,0,0,/16</p> <p>1,0,1,spare (defaults to /16)</p> <p>1,1,0,spare (defaults to /16)</p> <p>1,1,1,spare (defaults to /16)</p> <p>Note:</p> <p>[1] The CPU Timer2s Clock sync logic detects an input clock edge when configured for any clock source other than SYSCLK and generates an appropriate clock pulse to the CPU timer2. If SYSCLK is approximately the same or less then the input clock source, then the user would need to configure the pre-scale value such that SYSCLK is at least twice as fast as the pre-scaled value.</p> <p>Reset type: SYSRSn</p>
2-0	TMR2CLKSRCSEL	R/W	0h	<p>CPU Timer 2 Clock Source Select Bit: This bit selects the source for CPU Timer 2:</p> <p>000 =SYSCLK Selected (default on reset, pre-scale is bypassed)</p> <p>001 = INTOSC1</p> <p>010 = INTOSC2</p> <p>011 = XTAL</p> <p>100 = FLPUMPOSC</p> <p>101 = FOSCCLK</p> <p>110 = AUXPLLCLK (Reserved)</p> <p>111 = reserved</p> <p>Reset type: SYSRSn</p>

**3.14.4.25 RESCCLR Register (Offset = 7Eh) [reset = 0h]**

 RESCCLR is shown in [Figure 3-78](#) and described in [Table 3-80](#).

 Return to the [Summary Table](#).

Reset Cause Clear Register

**Figure 3-78. RESCCLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							SCCRESETn
R-0-0h							W1S-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	NMIWDRSn	WDRSn	XRSn	POR
R-0-0h			R-0-0h		W1S-0h	W1S-0h	W1S-0h

**Table 3-80. RESCCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-9	RESERVED	R-0	0h	Reserved
8	SCCRESETn	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn
7	RESERVED	R-0	0h	Reserved
6	RESERVED	W1S	0h	Reserved
5	RESERVED	W1S	0h	Reserved
4	RESERVED	R-0	0h	Reserved
3	NMIWDRSn	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn
2	WDRSn	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn
1	XRSn	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn



**Table 3-80. RESCCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	POR	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn

**3.14.4.26 RESC Register (Offset = 80h) [reset = X]**

RESC is shown in [Figure 3-79](#) and described in [Table 3-81](#).

Return to the [Summary Table](#).

Reset Cause register

**Figure 3-79. RESC Register**

31	30	29	28	27	26	25	24
DCON	XRSn_pin_status	RESERVED					
R-0h	R-X	R-0-0h					
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							SCCRESETn
R-0-0h							R-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	NMIWDRSn	WDRSn	XRSn	POR
R-0-0h			R-0-0h	R-0h	R-0h	R-1h	R-1h

**Table 3-81. RESC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	DCON	R	0h	Reading this bit provides the status of debugger connection to the C28x CPU. 0 : Debugger is not connected to the C28x CPU 1 : Debugger is connected to the C28x CPU  Notes: [1] This bit is connected to the DCON o/p signal of the C28x CPU Reset type: N/A
30	XRSn_pin_status	R	X	Reading this bit provides the current status of the XRSn pin. Reset value is reflective of the pin status. Reset type: N/A
29-9	RESERVED	R-0	0h	Reserved
8	SCCRESETn	R	0h	If this bit is set, indicates that the device was reset by SCCRESETn (fired by DCSM).  Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect. Reset type: POR
7	RESERVED	R-0	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R-0	0h	Reserved
3	NMIWDRSn	R	0h	If this bit is set, indicates that the device was reset by NMIWDRSn.  Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect.  To know the exact cause of NMI after the reset, software needs to read CPU1.NMISHDFLG registers Reset type: POR

**Table 3-81. RESC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	WDRSn	R	0h	<p>If this bit is set, indicates that the device was reset by WDRSn.</p> <p>Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect.</p> <p>Note: [1] A bit inside WD module also provides the same information. This bit is present to keep things consistent. This register is a one-stop shop for the software to know the reset cause for the C28x core. Reset type: POR</p>
1	XRSn	R	1h	<p>If this bit is set, indicates that the device was reset by XRSn.</p> <p>Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect. Reset type: POR</p>
0	POR	R	1h	<p>If this bit is set, indicates that the device was reset by POR.</p> <p>Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect. Reset type: POR</p>

### 3.14.5 CPUTIMER\_REGS Registers

Table 3-82 lists the CPUTIMER\_REGS registers. All register offset addresses not listed in Table 3-82 should be considered as reserved locations and the register contents should not be modified.

**Table 3-82. CPUTIMER\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	TIM	CPU-Timer, Counter Register		<a href="#">Go</a>
2h	PRD	CPU-Timer, Period Register		<a href="#">Go</a>
4h	TCR	CPU-Timer, Control Register		<a href="#">Go</a>
6h	TPR	CPU-Timer, Prescale Register		<a href="#">Go</a>
7h	TPRH	CPU-Timer, Prescale Register High		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-83 shows the codes that are used for access types in this section.

**Table 3-83. CPUTIMER\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
W1C	W 1C	Write 1 to clear
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.5.1 TIM Register (Offset = 0h) [reset = FFFFh]

TIM is shown in [Figure 3-80](#) and described in [Table 3-84](#).

Return to the [Summary Table](#).

CPU-Timer, Counter Register

**Figure 3-80. TIM 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
MSW																LSW															
R/W-0h																R/W-FFFFh															

**Table 3-84. TIM Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	MSW	R/W	0h	CPU-Timer Counter Registers The TIMH register holds the high 16 bits of the current 32-bit count of the timer. The TIMH:TIM decrements by one every (TDDR:1) clock cycles, where TDDR:TDDR is the timer prescale divideddown value. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRD:PRD registers. The timer interrupt (TINT) signal is generated. Reset type: SYSRSn
15-0	LSW	R/W	FFFFh	CPU-Timer Counter Registers The TIM register holds the low 16 bits of the current 32-bit count of the timer. The TIMH:TIM decrements by one every (TDDR:TDDR+1) clock cycles, where TDDR:TDDR is the timer prescale divideddown value. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRD:PRD registers. The timer interrupt (TINT) signal is generated. Reset type: SYSRSn

### 3.14.5.2 PRD Register (Offset = 2h) [reset = 0001FFFFh]

PRD is shown in [Figure 3-81](#) and described in [Table 3-85](#).

Return to the [Summary Table](#).

CPU-Timer, Period Register

**Figure 3-81. PRD 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
MSW																LSW															
R/W-1h																R/W-FFFFh															

**Table 3-85. PRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	MSW	R/W	1h	CPU-Timer Period Registers The PRDH register holds the high 16 bits of the 32-bit period. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers, at the start of the next timer input clock cycle (the output of the prescaler). The PRDH:PRD contents are also loaded into the TIMH:TIM when you set the timer reload bit (TRB) in the Timer Control Register (TCR). Reset type: SYSRSn
15-0	LSW	R/W	FFFFh	CPU-Timer Period Registers The PRD register holds the low 16 bits of the 32-bit period. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers, at the start of the next timer input clock cycle (the output of the prescaler). The PRDH:PRD contents are also loaded into the TIMH:TIM when you set the timer reload bit (TRB) in the Timer Control Register (TCR). Reset type: SYSRSn

### 3.14.5.3 TCR Register (Offset = 4h) [reset = 1h]

TCR is shown in [Figure 3-82](#) and described in [Table 3-86](#).

Return to the [Summary Table](#).

CPU-Timer, Control Register

**Figure 3-82. TCR Register**

15		14		13		12		11		10		9		8	
TIF		TIE		RESERVED				FREE		SOFT		RESERVED			
R/W1C-0h		R/W-0h		R-0h				R/W-0h		R/W-0h		R-0h			
7		6		5		4		3		2		1		0	
RESERVED				TRB		TSS		RESERVED							
R-0h				R/W-0h		R/W-0h		R-1h							

**Table 3-86. TCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	TIF	R/W1C	0h	<p>CPU-Timer Overflow Flag.</p> <p>TIF indicates whether a timer overflow has happened since TIF was last cleared. TIF is not cleared automatically and does not need to be cleared to enable the next timer interrupt.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The CPU-Timer has not decremented to zero. Writes of 0 are ignored.</p> <p>1h (R/W) = This flag gets set when the CPU-timer decrements to zero.</p> <p>Writing a 1 to this bit clears the flag.</p>
14	TIE	R/W	0h	<p>CPU-Timer Interrupt Enable.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The CPU-Timer interrupt is disabled.</p> <p>1h (R/W) = The CPU-Timer interrupt is enabled. If the timer decrements to zero, and TIE is set, the timer asserts its interrupt request.</p>
13-12	RESERVED	R	0h	Reserved
11	FREE	R/W	0h	<p>If the FREE bit is set to 1, then, upon a software breakpoint, the timer continues to run. If FREE is 0, then the SOFT bit controls the emulation behavior.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Stop after the next decrement of the TIMH:TIM (hard stop)</p> <p>1h (R/W) = Stop after the TIMH:TIM decrements to 0 (soft stop)</p> <p>In the SOFT STOP mode, the timer generates an interrupt before shutting down (since reaching 0 is the interrupt causing condition).</p> <p>2h (R/W) = Free run</p> <p>3h (R/W) = Free run</p>
10	SOFT	R/W	0h	<p>If the FREE bit is set to 1, then, upon a software breakpoint, the timer continues to run (that is, free runs). In this case, SOFT is a don't care. But if FREE is 0, then SOFT takes effect. In this case, if SOFT = 0, the timer halts the next time the TIMH:TIM decrements. If the SOFT bit is 1, then the timer halts when the TIMH:TIM has decremented to zero.</p> <p>Reset type: SYSRSn</p>
9-6	RESERVED	R	0h	Reserved

**Table 3-86. TCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	TRB	R/W	0h	Timer reload Reset type: SYSRSn 0h (R/W) = The TRB bit is always read as zero. Writes of 0 are ignored. 1h (R/W) = When you write a 1 to TRB, the TIMH:TIM is loaded with the value in the PRDH:PRD, and the prescaler counter (PSCH:PSC) is loaded with the value in the timer dividedown register (TDDR:TDDB).
4	TSS	R/W	0h	CPU-Timer stop status bit. TSS is a 1-bit flag that stops or starts the CPU-timer. Reset type: SYSRSn 0h (R/W) = Reads of 0 indicate the CPU-timer is running. To start or restart the CPU-timer, set TSS to 0. At reset, TSS is cleared to 0 and the CPU-timer immediately starts. 1h (R/W) = Reads of 1 indicate that the CPU-timer is stopped. To stop the CPU-timer, set TSS to 1.
3-0	RESERVED	R	1h	Reserved



### 3.14.5.4 TPR Register (Offset = 6h) [reset = 0h]

TPR is shown in [Figure 3-83](#) and described in [Table 3-87](#).

Return to the [Summary Table](#).

CPU-Timer, Prescale Register

**Figure 3-83. TPR Register**

15	14	13	12	11	10	9	8
PSC							
R-0h							
7	6	5	4	3	2	1	0
TDDR							
R/W-0h							

**Table 3-87. TPR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	PSC	R	0h	<p>CPU-Timer Prescale Counter.</p> <p>These bits hold the current prescale count for the timer. For every timer clock source cycle that the PSCH:PSC value is greater than 0, the PSCH:PSC decrements by one. One timer clock (output of the timer prescaler) cycle after the PSCH:PSC reaches 0, the PSCH:PSC is loaded with the contents of the TDDRH:TDDR, and the timer counter register (TIMH:TIM) decrements by one. The PSCH:PSC is also reloaded whenever the timer reload bit (TRB) is set by software. The PSCH:PSC can be checked by reading the register, but it cannot be set directly. It must get its value from the timer divide-down register (TDDRH:TDDR). At reset, the PSCH:PSC is set to 0.</p> <p>Reset type: SYSRSn</p>
7-0	TDDR	R/W	0h	<p>CPU-Timer Divide-Down.</p> <p>Every (TDDRH:TDDR + 1) timer clock source cycles, the timer counter register (TIMH:TIM) decrements by one. At reset, the TDDRH:TDDR bits are cleared to 0. To increase the overall timer count by an integer factor, write this factor minus one to the TDDRH:TDDR bits. When the prescaler counter (PSCH:PSC) value is 0, one timer clock source cycle later, the contents of the TDDRH:TDDR reload the PSCH:PSC, and the TIMH:TIM decrements by one. TDDRH:TDDR also reloads the PSCH:PSC whenever the timer reload bit (TRB) is set by software.</p> <p>Reset type: SYSRSn</p>

### 3.14.5.5 TPRH Register (Offset = 7h) [reset = 0h]

TPRH is shown in [Figure 3-84](#) and described in [Table 3-88](#).

Return to the [Summary Table](#).

CPU-Timer, Prescale Register High

**Figure 3-84. TPRH Register**

15	14	13	12	11	10	9	8
PSCH							
R-0h							
7	6	5	4	3	2	1	0
TDDRH							
R/W-0h							

**Table 3-88. TPRH Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	PSCH	R	0h	See description of TIMERxTPR. Reset type: SYSRSn
7-0	TDDRH	R/W	0h	See description of TIMERxTPR. Reset type: SYSRSn

### 3.14.6 DCSM\_BANK0\_Z1\_REGS Registers

Table 3-89 lists the DCSM\_BANK0\_Z1\_REGS registers. All register offset addresses not listed in Table 3-89 should be considered as reserved locations and the register contents should not be modified.

**Table 3-89. DCSM\_BANK0\_Z1\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	B0_Z1_LINKPOINTER	Zone 1 Link Pointer for flash BANK0		<a href="#">Go</a>
2h	Z1_OTPSECLOCK	Zone 1 OTP Secure JTAG lock		<a href="#">Go</a>
4h	Z1_BOOTDEF_HIGH	Boot definition (high 32bit)		<a href="#">Go</a>
6h	B0_Z1_LINKPOINTERERR	Link Pointer Error for flash BANK0		<a href="#">Go</a>
8h	Z1_BOOTPIN_CONFIG	Boot Pin Configuration		<a href="#">Go</a>
Ah	Z1_GPREG2	Zone1 General Purpose Register-2		<a href="#">Go</a>
Ch	Z1_BOOTDEF_LOW	Boot definition (low 32bit)		<a href="#">Go</a>
10h	Z1_CSMKEY0	Zone 1 CSM Key 0		<a href="#">Go</a>
12h	Z1_CSMKEY1	Zone 1 CSM Key 1		<a href="#">Go</a>
14h	Z1_CSMKEY2	Zone 1 CSM Key 2		<a href="#">Go</a>
16h	Z1_CSMKEY3	Zone 1 CSM Key 3		<a href="#">Go</a>
19h	Z1_CR	Zone 1 CSM Control Register		<a href="#">Go</a>
1Ah	B0_Z1_GRABSECTR	Zone 1 Grab Flash BANK0 Sectors Register		<a href="#">Go</a>
1Ch	Z1_GRABRAMR	Zone 1 Grab RAM Blocks Register		<a href="#">Go</a>
1Eh	B0_Z1_EXEONLYSECTR	Zone 1 Flash BANK0 Execute_Only Sector Register		<a href="#">Go</a>
20h	Z1_EXEONLYRAMR	Zone 1 RAM Execute_Only Block Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-90 shows the codes that are used for access types in this section.

**Table 3-90. DCSM\_BANK0\_Z1\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.6.1 B0\_Z1\_LINKPOINTER Register (Offset = 0h) [reset = E000000h]

B0\_Z1\_LINKPOINTER is shown in [Figure 3-85](#) and described in [Table 3-91](#).

Return to the [Summary Table](#).

Zone 1 Link Pointer for flash BANK0

**Figure 3-85. B0\_Z1\_LINKPOINTER Register**

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

**Table 3-91. B0\_Z1\_LINKPOINTER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	7h	Reserved
28-0	LINKPOINTER	R	0h	This is resolved Link-Pointer for Zone1 zone select block USER OTP of Flash BANK0. This is generated by using three physical Link-Pointer values loaded from OTP in Flash BANK0. Reset type: SYSRSn

**3.14.6.2 Z1\_OTPSECLOCK Register (Offset = 2h) [reset = Fh]**

Z1\_OTPSECLOCK is shown in [Figure 3-86](#) and described in [Table 3-92](#).

Return to the [Summary Table](#).

Zone 1 OTP Secure JTAG lock

**Figure 3-86. Z1\_OTPSECLOCK 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				CRCLOCK				PSWDLOCK				RESERVED			
R-0h				R-0h				R-0h							

**Table 3-92. Z1\_OTPSECLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
11-8	CRCLOCK	R	0h	Value in this field gets loaded from Z1OTP_CRCLOCK[3:0] when a read is issued to address location of Z1OTP_CRCLOCK in OTP.  1111 : VCU has ability to calculate CRC on secure memories. Other Value : VCU does not have ability to calculate CRC on secure memories.  Reset type: XRSn
7-4	PSWDLOCK	R	0h	Value in this field gets loaded from Z1OTP_PSWDLOCK[3:0] when a read is issued to address location of Z1OTP_PSWDLOCK in OTP.  1111 : CSM password locations in OTP are not protected and can be read from debugger as well as code running from anywhere. Other Value : CSM password locations in OTP are protected and can not be read without unlocking Zone1.  Reset type: XRSn
3-0	RESERVED	R	Fh	Reserved

### 3.14.6.3 Z1\_BOOTDEF\_HIGH Register (Offset = 4h) [reset = 0h]

Z1\_BOOTDEF\_HIGH is shown in [Figure 3-87](#) and described in [Table 3-93](#).

Return to the [Summary Table](#).

Boot definition (high 32bit)

**Figure 3-87. Z1\_BOOTDEF\_HIGH 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
BOOTDEF_HIGH																															
R-0h																															

**Table 3-93. Z1\_BOOTDEF\_HIGH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BOOTDEF_HIGH	R	0h	Refer ROM Code and Peripheral Booting section of TRM. Reset type: SYSRSn

### 3.14.6.4 B0\_Z1\_LINKPOINTERERR Register (Offset = 6h) [reset = 0h]

B0\_Z1\_LINKPOINTERERR is shown in [Figure 3-88](#) and described in [Table 3-94](#).

Return to the [Summary Table](#).

Link Pointer Error for flash BANK0

**Figure 3-88. B0\_Z1\_LINKPOINTERERR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED			Z1_LINKPOINTERERR												
R-0-0h			R-0h												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z1_LINKPOINTERERR															
R-0h															

**Table 3-94. B0\_Z1\_LINKPOINTERERR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R-0	0h	Reserved
28-0	Z1_LINKPOINTERERR	R	0h	<p>These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded from USER OTP in Flash BANK0</p> <p>0 : No Error.</p> <p>Other : Error on bit positions which is set to 1.</p> <p>Reset type: SYSRSn</p>

### 3.14.6.5 Z1\_BOOTPIN\_CONFIG Register (Offset = 8h) [reset = 0h]

Z1\_BOOTPIN\_CONFIG is shown in [Figure 3-89](#) and described in [Table 3-95](#).

Return to the [Summary Table](#).

Boot Pin Configuration

**Figure 3-89. Z1\_BOOTPIN\_CONFIG 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
BOOTPIN_CONFIG																															
R-0h																															

**Table 3-95. Z1\_BOOTPIN\_CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BOOTPIN_CONFIG	R	0h	Refer ROM Code and Peripheral Booting section of TRM. Reset type: SYSRSn



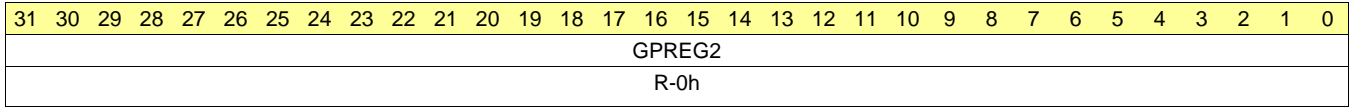
**3.14.6.6 Z1\_GPREG2 Register (Offset = Ah) [reset = 0h]**

Z1\_GPREG2 is shown in [Figure 3-90](#) and described in [Table 3-96](#).

Return to the [Summary Table](#).

Zone1 General Purpose Register-2

**Figure 3-90. Z1\_GPREG2 Register**



**Table 3-96. Z1\_GPREG2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GPREG2	R	0h	Refer ROM Code and Peripheral Booting section of TRM. Reset type: SYSRSn

### 3.14.6.7 Z1\_BOOTDEF\_LOW Register (Offset = Ch) [reset = 0h]

Z1\_BOOTDEF\_LOW is shown in [Figure 3-91](#) and described in [Table 3-97](#).

Return to the [Summary Table](#).

Boot definition (low 32bit)

**Figure 3-91. Z1\_BOOTDEF\_LOW 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
BOOTDEF_LOW																															
R-0h																															

**Table 3-97. Z1\_BOOTDEF\_LOW Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BOOTDEF_LOW	R	0h	Refer ROM Code and Peripheral Booting section of TRM. Reset type: SYSRSn

### 3.14.6.8 Z1\_CSMKEY0 Register (Offset = 10h) [reset = 0h]

Z1\_CSMKEY0 is shown in [Figure 3-92](#) and described in [Table 3-98](#).

Return to the [Summary Table](#).

Zone 1 CSM Key 0

**Figure 3-92. Z1\_CSMKEY0 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
Z1_CSMKEY0																															
R-0h																															

**Table 3-98. Z1\_CSMKEY0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1_CSMKEY0	R	0h	To unlock Zone1, user needs to write this register with exact value as Z1_CSMPSWD0, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.) Reset type: SYSRSn

### 3.14.6.9 Z1\_CSMKEY1 Register (Offset = 12h) [reset = 0h]

Z1\_CSMKEY1 is shown in [Figure 3-93](#) and described in [Table 3-99](#).

Return to the [Summary Table](#).

Zone 1 CSM Key 1

**Figure 3-93. Z1\_CSMKEY1 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
Z1_CSMKEY1																															
R-0h																															

**Table 3-99. Z1\_CSMKEY1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1_CSMKEY1	R	0h	To unlock Zone1, user needs to write this register with exact value as Z1_CSMPSWD1, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.) Reset type: SYSRSn

**3.14.6.10 Z1\_CSMKEY2 Register (Offset = 14h) [reset = 0h]**

Z1\_CSMKEY2 is shown in [Figure 3-94](#) and described in [Table 3-100](#).

Return to the [Summary Table](#).

Zone 1 CSM Key 2

**Figure 3-94. Z1\_CSMKEY2 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
Z1_CSMKEY2																															
R-0h																															

**Table 3-100. Z1\_CSMKEY2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1_CSMKEY2	R	0h	To unlock Zone1, user needs to write this register with exact value as Z1_CSMPSWD2, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.) Reset type: SYSRSn

**3.14.6.11 Z1\_CSMKEY3 Register (Offset = 16h) [reset = 0h]**

Z1\_CSMKEY3 is shown in [Figure 3-95](#) and described in [Table 3-101](#).

Return to the [Summary Table](#).

Zone 1 CSM Key 3

**Figure 3-95. Z1\_CSMKEY3 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
Z1_CSMKEY3																															
R-0h																															

**Table 3-101. Z1\_CSMKEY3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1_CSMKEY3	R	0h	To unlock Zone1, user needs to write this register with exact value as Z1_CSMPSWD3, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.) Reset type: SYSRSn

**3.14.6.12 Z1\_CR Register (Offset = 19h) [reset = 8h]**

Z1\_CR is shown in [Figure 3-96](#) and described in [Table 3-102](#).

Return to the [Summary Table](#).

Zone 1 CSM Control Register

**Figure 3-96. Z1\_CR Register**

15	14	13	12	11	10	9	8
FORCESEC		RESERVED					
R-0/W-0h		R-0h					
7	6	5	4	3	2	1	0
RESERVED	ARMED	UNSECURE	ALLONE	ALLZERO	RESERVED		
	R-0h	R-0h	R-0h	R-1h	R-0h		

**Table 3-102. Z1\_CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	FORCESEC	R-0/W	0h	A write '1' to this fields resets the state of zone. If zone is unlocked, it'll lock(secure) the zone and also resets all the bits in this register. Reset type: SYSRSn
14-8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	ARMED	R	0h	0 : Dummy read to CSM Password locations in USER OTP has not been performed. 1 : Dummy read to CSM Password locations in USER OTP has been performed. Reset type: SYSRSn
5	UNSECURE	R	0h	Indiacates the state of Zone1. 0 : Zone is in lock(secure) state. 1 : Zone is in unlock(unsecure) state. Reset type: SYSRSn
4	ALLONE	R	0h	Indicates the state of CSM passwords. 0 : Zone CSM Passwords are not all ones. 1 : Zone CSM Passwords are all ones and device is permanently blocked. Reset type: SYSRSn
3	ALLZERO	R	1h	Indicates the state of CSM passowrds. 0 : CSM Passwords are not all zeros. 1 : CSM Passwords are all zero and device is permanently locked. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

**3.14.6.13 B0\_Z1\_GRABSECTR Register (Offset = 1Ah) [reset = 0h]**

B0\_Z1\_GRABSECTR is shown in [Figure 3-97](#) and described in [Table 3-103](#).

Return to the [Summary Table](#).

Zone 1 Grab Flash BANK0 Sectors Register

**Figure 3-97. B0\_Z1\_GRABSECTR Register**

31	30	29	28	27	26	25	24
GRAB_SECT15		GRAB_SECT14		GRAB_SECT13		GRAB_SECT12	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECT11		GRAB_SECT10		GRAB_SECT9		GRAB_SECT8	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECT7		GRAB_SECT6		GRAB_SECT5		GRAB_SECT4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_SECT3		GRAB_SECT2		GRAB_SECT1		GRAB_SECT0	
R-0h		R-0h		R-0h		R-0h	

**Table 3-103. B0\_Z1\_GRABSECTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT15	R	0h	Value in this field gets loaded from B0_Z1OTP_GRABSECTR[31:30] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 15 is inaccessible. 01 : Request to allocate Flash Sector 15 to Zone1. 10 : No request for Flash Sector 15 11 : No request for Flash Sector 15 when this zone is UNLOCKED. Else Flash Sector 15 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn
29-28	GRAB_SECT14	R	0h	Value in this field gets loaded from B0_Z1OTP_GRABSECTR[29:28] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 14 is inaccessible. 01 : Request to allocate Flash Sector 14 to Zone1. 10 : No request for Flash Sector 14 11 : No request for Flash Sector 14 when this zone is UNLOCKED. Else Flash Sector 14 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn
27-26	GRAB_SECT13	R	0h	Value in this field gets loaded from B0_Z1OTP_GRABSECTR[27:26] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 13 is inaccessible. 01 : Request to allocate Flash Sector 13 to Zone1. 10 : No request for Flash Sector 13 11 : No request for Flash Sector 13 when this zone is UNLOCKED. Else Flash Sector 13 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn



**Table 3-103. B0\_Z1\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT12	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_GRABSECTR[25:24] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 12 is inaccessible.            01 : Request to allocate Flash Sector 12 to Zone1.            10 : No request for Flash Sector 12            11 : No request for Flash Sector 12 when this zone is UNLOCKED. Else Flash Sector 12 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
23-22	GRAB_SECT11	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_GRABSECTR[23:22] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 11 is inaccessible.            01 : Request to allocate Flash Sector 11 to Zone1.            10 : No request for Flash Sector 11            11 : No request for Flash Sector 11 when this zone is UNLOCKED. Else Flash Sector 11 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
21-20	GRAB_SECT10	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_GRABSECTR[21:20] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 10 is inaccessible.            01 : Request to allocate Flash Sector 10 to Zone1.            10 : No request for Flash Sector 10            11 : No request for Flash Sector 10 when this zone is UNLOCKED. Else Flash Sector 10 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
19-18	GRAB_SECT9	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_GRABSECTR[19:18] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 9 is inaccessible.            01 : Request to allocate Flash Sector 9 to Zone1.            10 : No request for Flash Sector 9            11 : No request for Flash Sector 9 when this zone is UNLOCKED. Else Flash Sector 9 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
17-16	GRAB_SECT8	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_GRABSECTR[17:16] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 8 is inaccessible.            01 : Request to allocate Flash Sector 8 to Zone1.            10 : No request for Flash Sector 8            11 : No request for Flash sector 8 when this zone is UNLOCKED. Else Flash sector 8 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>

**Table 3-103. B0\_Z1\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15-14	GRAB_SECT7	R	0h	Value in this field gets loaded from B0_Z1OTP_GRABSECTR[15:14] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 7 is inaccessible. 01 : Request to allocate Flash Sector 7 to Zone1. 10 : No request for Flash Sector 7 11 : No request for Flash Sector 7 when this zone is UNLOCKED. Else Flash Sector 7 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
13-12	GRAB_SECT6	R	0h	Value in this field gets loaded from B0_Z1OTP_GRABSECTR[13:12] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 6 is inaccessible. 01 : Request to allocate Flash Sector 6 to Zone1. 10 : No request for Flash Sector 6 11 : No request for Flash Sector 6 when this zone is UNLOCKED. Else Flash Sector 6 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_SECT5	R	0h	Value in this field gets loaded from B0_Z1OTP_GRABSECTR[11:10] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 5 is inaccessible. 01 : Request to allocate Flash Sector 5 to Zone1. 10 : No request for Flash Sector 5 11 : No request for Flash Sector 5 when this zone is UNLOCKED. Else Flash Sector 5 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
9-8	GRAB_SECT4	R	0h	Value in this field gets loaded from B0_Z1OTP_GRABSECTR[9:8] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 4 is inaccessible. 01 : Request to allocate Flash Sector 4 to Zone1. 10 : No request for Flash Sector 4 11 : No request for Flash Sector 4 when this zone is UNLOCKED. Else Flash Sector 4 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	GRAB_SECT3	R	0h	Value in this field gets loaded from B0_Z1OTP_GRABSECTR[7:6] when a read is issued to address location of B0_Z1OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 3 is inaccessible. 01 : Request to allocate Flash Sector 3 to Zone1. 10 : No request for Flash Sector 3 11 : No request for Flash Sector 3 when this zone is UNLOCKED. Else Flash Sector 3 is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 3-103. B0\_Z1\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-4	GRAB_SECT2	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_GRABSECT[5:4] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 2 is inaccessible.            01 : Request to allocate Flash Sector 2 to Zone1.            10 : No request for Flash Sector 2            11 : No request for Flash Sector 2 when this zone is UNLOCKED. Else Flash Sector 2 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
3-2	GRAB_SECT1	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_GRABSECT[3:2] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 1 is inaccessible.            01 : Request to allocate Flash Sector 1 to Zone1.            10 : No request for Flash Sector 1            11 : No request for Flash sector 1 when this zone is UNLOCKED. Else Flash sector 1 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
1-0	GRAB_SECT0	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_GRABSECT[1:0] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 0 is inaccessible.            01 : Request to allocate Flash Sector 0 to Zone1.            10 : No request for Flash Sector 0            11 : No request for Flash Sector 0 when this zone is UNLOCKED. Else Flash Sector 0 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>

**3.14.6.14 Z1\_GRABRAMR Register (Offset = 1Ch) [reset = 0h]**

 Z1\_GRABRAMR is shown in [Figure 3-98](#) and described in [Table 3-104](#).

 Return to the [Summary Table](#).

Zone 1 Grab RAM Blocks Register

**Figure 3-98. Z1\_GRABRAMR 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
GRAB_RAM7		GRAB_RAM6		GRAB_RAM5		GRAB_RAM4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_RAM3		GRAB_RAM2		GRAB_RAM1		GRAB_RAM0	
R-0h		R-0h		R-0h		R-0h	

**Table 3-104. Z1\_GRABRAMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	GRAB_RAM7	R	0h	Value in this field gets loaded from Z1OTP_GRABRAM[15:14] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.  00 : Invalid. LS7 RAM is inaccessible. 01 : Request to allocate LS7 RAM to Zone1. 10 : No request for LS7 RAM 11 : No request for LS7 RAM when this zone is UNLOCKED. Else LS7 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
13-12	GRAB_RAM6	R	0h	Value in this field gets loaded from Z1OTP_GRABRAM[13:12] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.  00 : Invalid. LS6 RAM is inaccessible. 01 : Request to allocate LS6 RAM to Zone1. 10 : No request for LS6 RAM 11 : No request for LS6 RAM when this zone is UNLOCKED. Else LS6 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_RAM5	R	0h	Value in this field gets loaded from Z1OTP_GRABRAM[11:10] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.  00 : Invalid. LS5 RAM is inaccessible. 01 : Request to allocate LS5 RAM to Zone1. 10 : No request for LS5 RAM 11 : No request for LS5 RAM when this zone is UNLOCKED. Else LS5 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 3-104. Z1\_GRABRAMR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	GRAB_RAM4	R	0h	<p>Value in this field gets loaded from Z1OTP_GRABRAM[9:8] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.</p> <p>00 : Invalid. LS4 RAM is inaccessible.            01 : Request to allocate LS4 RAM to Zone1.            10 : No request for LS4 RAM            11 : No request for LS4 RAM when this zone is UNLOCKED. Else LS4 RAM is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
7-6	GRAB_RAM3	R	0h	<p>Value in this field gets loaded from Z1OTP_GRABRAM[7:6] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.</p> <p>00 : Invalid. LS3 RAM is inaccessible.            01 : Request to allocate LS3 RAM to Zone1.            10 : No request for LS3 RAM            11 : No request for LS3 RAM when this zone is UNLOCKED. Else LS3 RAM is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
5-4	GRAB_RAM2	R	0h	<p>Value in this field gets loaded from Z1OTP_GRABRAM[5:4] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.</p> <p>00 : Invalid. LS2 RAM is inaccessible.            01 : Request to allocate LS2 RAM to Zone1.            10 : No request for LS2 RAM            11 : No request for LS2 RAM when this zone is UNLOCKED. Else LS2 RAM is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
3-2	GRAB_RAM1	R	0h	<p>Value in this field gets loaded from Z1OTP_GRABRAM[3:2] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.</p> <p>00 : Invalid. LS1 RAM is inaccessible.            01 : Request to allocate LS1 RAM to Zone1.            10 : No request for LS1 RAM            11 : No request for LS1 RAM when this zone is UNLOCKED. Else LS1 RAM is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
1-0	GRAB_RAM0	R	0h	<p>Value in this field gets loaded from Z1OTP_GRABRAM[1:0] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.</p> <p>00 : Invalid. LS0 RAM is inaccessible.            01 : Request to allocate LS0 RAM to Zone1.            10 : No request for LS0 RAM            11 : No request for LS0 RAM when this zone is UNLOCKED. Else LS0 RAM is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>

**3.14.6.15 B0\_Z1\_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]**

B0\_Z1\_EXEONLYSECTR is shown in [Figure 3-99](#) and described in [Table 3-105](#).

Return to the [Summary Table](#).

Zone 1 Flash BANK0 Execute\_Only Sector Register

**Figure 3-99. B0\_Z1\_EXEONLYSECTR 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
EXEONLY_SE CT15	EXEONLY_SE CT14	EXEONLY_SE CT13	EXEONLY_SE CT12	EXEONLY_SE CT11	EXEONLY_SE CT10	EXEONLY_SE CT9	EXEONLY_SE CT8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
EXEONLY_SE CT7	EXEONLY_SE CT6	EXEONLY_SE CT5	EXEONLY_SE CT4	EXEONLY_SE CT3	EXEONLY_SE CT2	EXEONLY_SE CT1	EXEONLY_SE CT0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-105. B0\_Z1\_EXEONLYSECTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	EXEONLY_SECT15	R	0h	Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[15:15] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 15 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 15 (only if it's allocated to Zone1)  Reset type: SYSRSn
14	EXEONLY_SECT14	R	0h	Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[14:14] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 14 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 14 (only if it's allocated to Zone1)  Reset type: SYSRSn
13	EXEONLY_SECT13	R	0h	Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[13:13] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 13 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 13 (only if it's allocated to Zone1)  Reset type: SYSRSn

**Table 3-105. B0\_Z1\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	EXEONLY_SECT12	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[12:12] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 12 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 12 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
11	EXEONLY_SECT11	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[11:11] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 11 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 11 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
10	EXEONLY_SECT10	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[10:10] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 10 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 10 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
9	EXEONLY_SECT9	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[9:9] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 9 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 9 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
8	EXEONLY_SECT8	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[8:8] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 8 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 8 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
7	EXEONLY_SECT7	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[7:7] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 7 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 7 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>

**Table 3-105. B0\_Z1\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	EXEONLY_SECT6	R	0h	Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[6:6] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 6 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 6 (only if it's allocated to Zone1) Reset type: SYSRSn
5	EXEONLY_SECT5	R	0h	Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[5:5] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 5 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 5 (only if it's allocated to Zone1) Reset type: SYSRSn
4	EXEONLY_SECT4	R	0h	Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[4:4] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 4 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 4 (only if it's allocated to Zone1) Reset type: SYSRSn
3	EXEONLY_SECT3	R	0h	Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[3:3] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 3 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 3 (only if it's allocated to Zone1) Reset type: SYSRSn
2	EXEONLY_SECT2	R	0h	Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[2:2] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 2 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 2 (only if it's allocated to Zone1) Reset type: SYSRSn
1	EXEONLY_SECT1	R	0h	Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[1:1] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 1 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 1 (only if it's allocated to Zone1) Reset type: SYSRSn



**Table 3-105. B0\_Z1\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	EXEONLY_SECT0	R	0h	<p>Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[0:0] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 0 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 0 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>

**3.14.6.16 Z1\_EXEONLYRAMR Register (Offset = 20h) [reset = 0h]**

 Z1\_EXEONLYRAMR is shown in [Figure 3-100](#) and described in [Table 3-106](#).

 Return to the [Summary Table](#).

Zone 1 RAM Execute\_Only Block Register

**Figure 3-100. Z1\_EXEONLYRAMR 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
EXEONLY_RA M7	EXEONLY_RA M6	EXEONLY_RA M5	EXEONLY_RA M4	EXEONLY_RA M3	EXEONLY_RA M2	EXEONLY_RA M1	EXEONLY_RA M0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-106. Z1\_EXEONLYRAMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	EXEONLY_RAM7	R	0h	Value in this field gets loaded from Z1OTP_EXEONLYRAM[7:7] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS7 RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for LS7 RAM (only if it's allocated to Zone1)  Reset type: SYSRSn
6	EXEONLY_RAM6	R	0h	Value in this field gets loaded from Z1OTP_EXEONLYRAM[6:6] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS6 RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for LS6 RAM (only if it's allocated to Zone1)  Reset type: SYSRSn
5	EXEONLY_RAM5	R	0h	Value in this field gets loaded from Z1OTP_EXEONLYRAM[5:5] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS5 RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for LS5 RAM (only if it's allocated to Zone1)  Reset type: SYSRSn

**Table 3-106. Z1\_EXEONLYRAMR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	EXEONLY_RAM4	R	0h	<p>Value in this field gets loaded from Z1OTP_EXEONLYRAM[4:4] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.</p> <p>0 : Execute-Only protection is enabled for LS4 RAM (only if it's allocated to Zone1)            1 : Execute-Only protection is disabled for LS4 RAM (only if it's allocated to Zone1)            Reset type: SYSRSn</p>
3	EXEONLY_RAM3	R	0h	<p>Value in this field gets loaded from Z1OTP_EXEONLYRAM[3:3] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.</p> <p>0 : Execute-Only protection is enabled for LS3 RAM (only if it's allocated to Zone1)            1 : Execute-Only protection is disabled for LS3 RAM (only if it's allocated to Zone1)            Reset type: SYSRSn</p>
2	EXEONLY_RAM2	R	0h	<p>Value in this field gets loaded from Z1OTP_EXEONLYRAM[2:2] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.</p> <p>0 : Execute-Only protection is enabled for LS2 RAM (only if it's allocated to Zone1)            1 : Execute-Only protection is disabled for LS2 RAM (only if it's allocated to Zone1)            Reset type: SYSRSn</p>
1	EXEONLY_RAM1	R	0h	<p>Value in this field gets loaded from Z1OTP_EXEONLYRAM[1:1] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.</p> <p>0 : Execute-Only protection is enabled for LS1 RAM (only if it's allocated to Zone1)            1 : Execute-Only protection is disabled for LS1 RAM (only if it's allocated to Zone1)            Reset type: SYSRSn</p>
0	EXEONLY_RAM0	R	0h	<p>Value in this field gets loaded from Z1OTP_EXEONLYRAM[0:0] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.</p> <p>0 : Execute-Only protection is enabled for LS0 RAM (only if it's allocated to Zone1)            1 : Execute-Only protection is disabled for LS0 RAM (only if it's allocated to Zone1)            Reset type: SYSRSn</p>

### 3.14.7 DCSM\_BANK0\_Z2\_REGS Registers

Table 3-107 lists the DCSM\_BANK0\_Z2\_REGS registers. All register offset addresses not listed in Table 3-107 should be considered as reserved locations and the register contents should not be modified.

**Table 3-107. DCSM\_BANK0\_Z2\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	B0_Z2_LINKPOINTER	Zone 2 Link Pointer for flash BANK0		<a href="#">Go</a>
2h	Z2_OTPSECLOCK	Zone 2 OTP Secure JTAG lock		<a href="#">Go</a>
6h	B0_Z2_LINKPOINTERERR	Link Pointer Error for flash BANK0		<a href="#">Go</a>
10h	Z2_CSMKEY0	Zone 2 CSM Key 0		<a href="#">Go</a>
12h	Z2_CSMKEY1	Zone 2 CSM Key 1		<a href="#">Go</a>
14h	Z2_CSMKEY2	Zone 2 CSM Key 2		<a href="#">Go</a>
16h	Z2_CSMKEY3	Zone 2 CSM Key 3		<a href="#">Go</a>
19h	Z2_CR	Zone 2 CSM Control Register		<a href="#">Go</a>
1Ah	B0_Z2_GRABSECTR	Zone 2 Grab Flash BANK0 Sectors Register		<a href="#">Go</a>
1Ch	Z2_GRABRAMR	Zone 2 Grab RAM Blocks Register		<a href="#">Go</a>
1Eh	B0_Z2_EXEONLYSECTR	Zone 2 Flash BANK0 Execute_Only Sector Register		<a href="#">Go</a>
20h	Z2_EXEONLYRAMR	Zone 2 RAM Execute_Only Block Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-108 shows the codes that are used for access types in this section.

**Table 3-108. DCSM\_BANK0\_Z2\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.7.1 B0\_Z2\_LINKPOINTER Register (Offset = 0h) [reset = E000000h]

B0\_Z2\_LINKPOINTER is shown in [Figure 3-101](#) and described in [Table 3-109](#).

Return to the [Summary Table](#).

Zone 2 Link Pointer for flash BANK0

**Figure 3-101. B0\_Z2\_LINKPOINTER Register**

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

**Table 3-109. B0\_Z2\_LINKPOINTER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	7h	Reserved
28-0	LINKPOINTER	R	0h	This is resolved Link-Pointer for Zone2 zone select block USER OTP of Flash BANK0. This is generated by using three physical Link-Pointer values loaded from OTP in Flash BANK0. Reset type: SYSRSn

**3.14.7.2 Z2\_OTPSECLOCK Register (Offset = 2h) [reset = Fh]**

Z2\_OTPSECLOCK is shown in [Figure 3-102](#) and described in [Table 3-110](#).

Return to the [Summary Table](#).

Zone 2 OTP Secure JTAG lock

**Figure 3-102. Z2\_OTPSECLOCK 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				CRCLOCK				PSWDLOCK				RESERVED			
R-0h				R-0h				R-0h							

**Table 3-110. Z2\_OTPSECLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
11-8	CRCLOCK	R	0h	Value in this field gets loaded from Z2_CRCLOCK[3:0] when a read is issued to address location of Z2OTP_CRCLOCK in OTP.  1111 : VCU has ability to calculate CRC on secure memories. Other Value : VCU doesn't have ability to calculate CRC on secure memories. Reset type: XRSn
7-4	PSWDLOCK	R	0h	Value in this field gets loaded from Z2_PSWDLOCK[3:0] when a read is issued to address location of Z2OTP_PSWDLOCK in OTP.  1111 : CSM password locations in OTP are not protected and can be read from debugger as well as code running from anywhere. Other Value : CSM password locations in OTP are protected and can't be read without unlocking CSM of that zone. Reset type: XRSn
3-0	RESERVED	R	Fh	Reserved

### 3.14.7.3 B0\_Z2\_LINKPOINTERERR Register (Offset = 6h) [reset = 0h]

B0\_Z2\_LINKPOINTERERR is shown in [Figure 3-103](#) and described in [Table 3-111](#).

Return to the [Summary Table](#).

Link Pointer Error for flash BANK0

**Figure 3-103. B0\_Z2\_LINKPOINTERERR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED			Z2_LINKPOINTERERR												
R-0-0h			R-0h												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z2_LINKPOINTERERR															
R-0h															

**Table 3-111. B0\_Z2\_LINKPOINTERERR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R-0	0h	Reserved
28-0	Z2_LINKPOINTERERR	R	0h	<p>These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded from USER OTP in Flash BANK0</p> <p>0 : No Error.</p> <p>Other : Error on bit positions which is set to 1.</p> <p>Reset type: SYSRSn</p>

### 3.14.7.4 Z2\_CSMKEY0 Register (Offset = 10h) [reset = 0h]

Z2\_CSMKEY0 is shown in [Figure 3-104](#) and described in [Table 3-112](#).

Return to the [Summary Table](#).

Zone 2 CSM Key 0

**Figure 3-104. Z2\_CSMKEY0 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
Z2_CSMKEY0																															
R-0h																															

**Table 3-112. Z2\_CSMKEY0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2_CSMKEY0	R	0h	To unlock Zone2 user needs to write this register with exact value as Z2_CSMPSWD0, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.) Reset type: SYSRSn



### 3.14.7.5 Z2\_CSMKEY1 Register (Offset = 12h) [reset = 0h]

Z2\_CSMKEY1 is shown in [Figure 3-105](#) and described in [Table 3-113](#).

Return to the [Summary Table](#).

Zone 2 CSM Key 1

**Figure 3-105. Z2\_CSMKEY1 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
Z2_CSMKEY1																															
R-0h																															

**Table 3-113. Z2\_CSMKEY1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2_CSMKEY1	R	0h	To unlock Zone2 user needs to write this register with exact value as Z2_CSMPSWD1, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.) Reset type: SYSRSn

### 3.14.7.6 Z2\_CSMKEY2 Register (Offset = 14h) [reset = 0h]

Z2\_CSMKEY2 is shown in [Figure 3-106](#) and described in [Table 3-114](#).

Return to the [Summary Table](#).

Zone 2 CSM Key 2

**Figure 3-106. Z2\_CSMKEY2 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
Z2_CSMKEY2																															
R-0h																															

**Table 3-114. Z2\_CSMKEY2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2_CSMKEY2	R	0h	To unlock Zone2 user needs to write this register with exact value as Z2_CSMPSWD2, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.) Reset type: SYSRSn

**3.14.7.7 Z2\_CSMKEY3 Register (Offset = 16h) [reset = 0h]**

Z2\_CSMKEY3 is shown in [Figure 3-107](#) and described in [Table 3-115](#).

Return to the [Summary Table](#).

Zone 2 CSM Key 3

**Figure 3-107. Z2\_CSMKEY3 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
Z2_CSMKEY3																															
R-0h																															

**Table 3-115. Z2\_CSMKEY3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2_CSMKEY3	R	0h	To unlock Zone2 user needs to write this register with exact value as Z2_CSMPSWD3, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.) Reset type: SYSRSn

**3.14.7.8 Z2\_CR Register (Offset = 19h) [reset = 8h]**

Z2\_CR is shown in [Figure 3-108](#) and described in [Table 3-116](#).

Return to the [Summary Table](#).

Zone 2 CSM Control Register

**Figure 3-108. Z2\_CR Register**

15	14	13	12	11	10	9	8
FORCESEC	RESERVED						
R-0/W-0h	R-0-0h						
7	6	5	4	3	2	1	0
RESERVED	ARMED	UNSECURE	ALLONE	ALLZERO	RESERVED		
	R-0h	R-0h	R-0h	R-1h	R-0h		

**Table 3-116. Z2\_CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	FORCESEC	R-0/W	0h	A write '1' to this fields resets the state of zone. If zone is unlocked, it'll lock(secure) the zone and also resets all the bits in this register. Reset type: SYSRSn
14-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	ARMED	R	0h	0 : Dummy read to CSM Password locations in USER OTP has not been performed. 1 : Dummy read to CSM Password locations in USER OTP has been performed. Reset type: SYSRSn
5	UNSECURE	R	0h	Indicates the state of Zone. 0 : Zone is in lock(secure) state. 1 : Zone is in unlock(unsecure) state. Reset type: SYSRSn
4	ALLONE	R	0h	Indicates the state of CSM passwords. 0 : Zone CSM Passwords are not all ones. 1 : Zone CSM Passwords are all ones and device is permanently blocked. Reset type: SYSRSn
3	ALLZERO	R	1h	Indicates the state of CSM passowrds. 0 : CSM Passwords are not all zeros. 1 : CSM Passwords are all zero and device is permanently locked. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

### 3.14.7.9 B0\_Z2\_GRABSECTR Register (Offset = 1Ah) [reset = 0h]

B0\_Z2\_GRABSECTR is shown in [Figure 3-109](#) and described in [Table 3-117](#).

Return to the [Summary Table](#).

Zone 2 Grab Flash BANK0 Sectors Register

**Figure 3-109. B0\_Z2\_GRABSECTR Register**

31	30	29	28	27	26	25	24
GRAB_SECT15		GRAB_SECT14		GRAB_SECT13		GRAB_SECT12	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECT11		GRAB_SECT10		GRAB_SECT9		GRAB_SECT8	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECT7		GRAB_SECT6		GRAB_SECT5		GRAB_SECT4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_SECT3		GRAB_SECT2		GRAB_SECT1		GRAB_SECT0	
R-0h		R-0h		R-0h		R-0h	

**Table 3-117. B0\_Z2\_GRABSECTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT15	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECTR[31:30] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 15 is inaccessible. 01 : Request to allocate Flash Sector 15 to Zone2. 10 : No request for Flash Sector 15 11 : No request for Flash Sector 15 when this zone is UNLOCKED. Else Flash Sector 15 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn
29-28	GRAB_SECT14	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECTR[29:28] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 14 is inaccessible. 01 : Request to allocate Flash Sector 14 to Zone2. 10 : No request for Flash Sector 14 11 : No request for Flash Sector 14 when this zone is UNLOCKED. Else Flash Sector 14 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn
27-26	GRAB_SECT13	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECTR[27:26] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 13 is inaccessible. 01 : Request to allocate Flash Sector 13 to Zone2. 10 : No request for Flash Sector 13 11 : No request for Flash Sector 13 when this zone is UNLOCKED. Else Flash Sector 13 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn

**Table 3-117. B0\_Z2\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT12	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECTR[25:24] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 12 is inaccessible. 01 : Request to allocate Flash Sector 12 to Zone2. 10 : No request for Flash Sector 12 11 : No request for Flash Sector 12 when this zone is UNLOCKED. Else Flash Sector 12 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
23-22	GRAB_SECT11	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECTR[23:22] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 11 is inaccessible. 01 : Request to allocate Flash Sector 11 to Zone2. 10 : No request for Flash Sector 11 11 : No request for Flash Sector 11 when this zone is UNLOCKED. Else Flash Sector 11 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
21-20	GRAB_SECT10	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECTR[21:20] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 10 is inaccessible. 01 : Request to allocate Flash Sector 10 to Zone2. 10 : No request for Flash Sector 10 11 : No request for Flash Sector 10 when this zone is UNLOCKED. Else Flash Sector 10 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
19-18	GRAB_SECT9	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECTR[19:18] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 9 is inaccessible. 01 : Request to allocate Flash Sector 9 to Zone2. 10 : No request for Flash Sector 9 11 : No request for Flash Sector 9 when this zone is UNLOCKED. Else Flash Sector 9 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
17-16	GRAB_SECT8	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECTR[17:16] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 8 is inaccessible. 01 : Request to allocate Flash Sector 8 to Zone2. 10 : No request for Flash Sector 8 11 : No request for Flash sector 8 when this zone is UNLOCKED. Else Flash sector 8 is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 3-117. B0\_Z2\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15-14	GRAB_SECT7	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_GRABSECTR[15:14] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 7 is inaccessible.            01 : Request to allocate Flash Sector 7 to Zone2.            10 : No request for Flash Sector 7            11 : No request for Flash Sector 7 when this zone is UNLOCKED. Else Flash Sector 7 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
13-12	GRAB_SECT6	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_GRABSECTR[13:12] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 6 is inaccessible.            01 : Request to allocate Flash Sector 6 to Zone2.            10 : No request for Flash Sector 6            11 : No request for Flash Sector 6 when this zone is UNLOCKED. Else Flash Sector 6 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
11-10	GRAB_SECT5	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_GRABSECTR[11:10] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 5 is inaccessible.            01 : Request to allocate Flash Sector 5 to Zone2.            10 : No request for Flash Sector 5            11 : No request for Flash Sector 5 when this zone is UNLOCKED. Else Flash Sector 5 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
9-8	GRAB_SECT4	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_GRABSECTR[9:8] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 4 is inaccessible.            01 : Request to allocate Flash Sector 4 to Zone2.            10 : No request for Flash Sector 4            11 : No request for Flash Sector 4 when this zone is UNLOCKED. Else Flash Sector 4 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
7-6	GRAB_SECT3	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_GRABSECTR[7:6] when a read is issued to address location of B0_Z2OTP_GRABSECTR in USER OTP of Flash BANK0.</p> <p>00 : Invalid. Flash Sector 3 is inaccessible.            01 : Request to allocate Flash Sector 3 to Zone2.            10 : No request for Flash Sector 3            11 : No request for Flash Sector 3 when this zone is UNLOCKED. Else Flash Sector 3 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>

**Table 3-117. B0\_Z2\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-4	GRAB_SECT2	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECT[5:4] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 2 is inaccessible. 01 : Request to allocate Flash Sector 2 to Zone2. 10 : No request for Flash Sector 2 11 : No request for Flash Sector 2 when this zone is UNLOCKED. Else Flash Sector 2 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
3-2	GRAB_SECT1	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECT[3:2] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 1 is inaccessible. 01 : Request to allocate Flash Sector 1 to Zone2. 10 : No request for Flash Sector 1 11 : No request for Flash sector 1 when this zone is UNLOCKED. Else Flash sector 1 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
1-0	GRAB_SECT0	R	0h	Value in this field gets loaded from B0_Z2OTP_GRABSECT[1:0] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.  00 : Invalid. Flash Sector 0 is inaccessible. 01 : Request to allocate Flash Sector 0 to Zone2. 10 : No request for Flash Sector 0 11 : No request for Flash Sector 0 when this zone is UNLOCKED. Else Flash Sector 0 is inaccessible if this zone is LOCKED. Reset type: SYSRSn



**3.14.7.10 Z2\_GRABRAMR Register (Offset = 1Ch) [reset = 0h]**

 Z2\_GRABRAMR is shown in [Figure 3-110](#) and described in [Table 3-118](#).

 Return to the [Summary Table](#).

Zone 2 Grab RAM Blocks Register

**Figure 3-110. Z2\_GRABRAMR 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
GRAB_RAM7		GRAB_RAM6		GRAB_RAM5		GRAB_RAM4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_RAM3		GRAB_RAM2		GRAB_RAM1		GRAB_RAM0	
R-0h		R-0h		R-0h		R-0h	

**Table 3-118. Z2\_GRABRAMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	GRAB_RAM7	R	0h	Value in this field gets loaded from Z2OTP_GRABRAM[15:14] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.  00 : Invalid. LS7 RAM is inaccessible. 01 : Request to allocate LS7 RAM to Zone2. 10 : No request for LS7 RAM 11 : No request for LS7 RAM when this zone is UNLOCKED. Else LS7 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
13-12	GRAB_RAM6	R	0h	Value in this field gets loaded from Z2OTP_GRABRAM[13:12] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.  00 : Invalid. LS6 RAM is inaccessible. 01 : Request to allocate LS6 RAM to Zone2. 10 : No request for LS6 RAM 11 : No request for LS6 RAM when this zone is UNLOCKED. Else LS6 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_RAM5	R	0h	Value in this field gets loaded from Z2OTP_GRABRAM[11:10] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.  00 : Invalid. LS5 RAM is inaccessible. 01 : Request to allocate LS5 RAM to Zone2. 10 : No request for LS5 RAM 11 : No request for LS5 RAM when this zone is UNLOCKED. Else LS5 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 3-118. Z2\_GRABRAMR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	GRAB_RAM4	R	0h	Value in this field gets loaded from Z2OTP_GRABRAM[9:8] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.  00 : Invalid. LS4 RAM is inaccessible. 01 : Request to allocate LS4 RAM to Zone2. 10 : No request for LS4 RAM 11 : No request for LS4 RAM when this zone is UNLOCKED. Else LS4 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	GRAB_RAM3	R	0h	Value in this field gets loaded from Z2OTP_GRABRAM[7:6] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.  00 : Invalid. LS3 RAM is inaccessible. 01 : Request to allocate LS3 RAM to Zone2. 10 : No request for LS3 RAM 11 : No request for LS3 RAM when this zone is UNLOCKED. Else LS3 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
5-4	GRAB_RAM2	R	0h	Value in this field gets loaded from Z2OTP_GRABRAM[5:4] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.  00 : Invalid. LS2 RAM is inaccessible. 01 : Request to allocate LS2 RAM to Zone2. 10 : No request for LS2 RAM 11 : No request for LS2 RAM when this zone is UNLOCKED. Else LS2 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
3-2	GRAB_RAM1	R	0h	Value in this field gets loaded from Z2OTP_GRABRAM[3:2] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.  00 : Invalid. LS1 RAM is inaccessible. 01 : Request to allocate LS1 RAM to Zone2. 10 : No request for LS1 RAM 11 : No request for LS1 RAM when this zone is UNLOCKED. Else LS1 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
1-0	GRAB_RAM0	R	0h	Value in this field gets loaded from Z2OTP_GRABRAM[1:0] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.  00 : Invalid. LS0 RAM is inaccessible. 01 : Request to allocate LS0 RAM to Zone2. 10 : No request for LS0 RAM 11 : No request for LS0 RAM when this zone is UNLOCKED. Else LS0 RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**3.14.7.11 B0\_Z2\_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]**

 B0\_Z2\_EXEONLYSECTR is shown in [Figure 3-111](#) and described in [Table 3-119](#).

 Return to the [Summary Table](#).

Zone 2 Flash BANK0 Execute\_Only Sector Register

**Figure 3-111. B0\_Z2\_EXEONLYSECTR 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
EXEONLY_SE CT15	EXEONLY_SE CT14	EXEONLY_SE CT13	EXEONLY_SE CT12	EXEONLY_SE CT11	EXEONLY_SE CT10	EXEONLY_SE CT9	EXEONLY_SE CT8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
EXEONLY_SE CT7	EXEONLY_SE CT6	EXEONLY_SE CT5	EXEONLY_SE CT4	EXEONLY_SE CT3	EXEONLY_SE CT2	EXEONLY_SE CT1	EXEONLY_SE CT0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-119. B0\_Z2\_EXEONLYSECTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	EXEONLY_SECT15	R	0h	Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[15:15] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 15 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 15 (only if it's allocated to Zone2)  Reset type: SYSRSn
14	EXEONLY_SECT14	R	0h	Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[14:14] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 14 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 14 (only if it's allocated to Zone2)  Reset type: SYSRSn
13	EXEONLY_SECT13	R	0h	Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[13:13] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 13 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 13 (only if it's allocated to Zone2)  Reset type: SYSRSn

**Table 3-119. B0\_Z2\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	EXEONLY_SECT12	R	0h	Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[12:12] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 12 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 12 (only if it's allocated to Zone2) Reset type: SYSRSn
11	EXEONLY_SECT11	R	0h	Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[11:11] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 11 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 11 (only if it's allocated to Zone2) Reset type: SYSRSn
10	EXEONLY_SECT10	R	0h	Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[10:10] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 10 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 10 (only if it's allocated to Zone2) Reset type: SYSRSn
9	EXEONLY_SECT9	R	0h	Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[9:9] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 9 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 9 (only if it's allocated to Zone2) Reset type: SYSRSn
8	EXEONLY_SECT8	R	0h	Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[8:8] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 8 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 8 (only if it's allocated to Zone2) Reset type: SYSRSn
7	EXEONLY_SECT7	R	0h	Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[7:7] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.  0 : Execute-Only protection is enabled for Flash Sector 7 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 7 (only if it's allocated to Zone2) Reset type: SYSRSn

**Table 3-119. B0\_Z2\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	EXEONLY_SECT6	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[6:6] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 6 (only if it's allocated to Zone2)            1 : Execute-Only protection is disabled for Flash Sector 6 (only if it's allocated to Zone2)            Reset type: SYSRSn</p>
5	EXEONLY_SECT5	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[5:5] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 5 (only if it's allocated to Zone2)            1 : Execute-Only protection is disabled for Flash Sector 5 (only if it's allocated to Zone2)            Reset type: SYSRSn</p>
4	EXEONLY_SECT4	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[4:4] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 4 (only if it's allocated to Zone2)            1 : Execute-Only protection is disabled for Flash Sector 4 (only if it's allocated to Zone2)            Reset type: SYSRSn</p>
3	EXEONLY_SECT3	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[3:3] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 3 (only if it's allocated to Zone2)            1 : Execute-Only protection is disabled for Flash Sector 3 (only if it's allocated to Zone2)            Reset type: SYSRSn</p>
2	EXEONLY_SECT2	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[2:2] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 2 (only if it's allocated to Zone2)            1 : Execute-Only protection is disabled for Flash Sector 2 (only if it's allocated to Zone2)            Reset type: SYSRSn</p>
1	EXEONLY_SECT1	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[1:1] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 1 (only if it's allocated to Zone2)            1 : Execute-Only protection is disabled for Flash Sector 1 (only if it's allocated to Zone2)            Reset type: SYSRSn</p>

**Table 3-119. B0\_Z2\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	EXEONLY_SECT0	R	0h	<p>Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[0:0] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 0 (only if it's allocated to Zone2)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 0 (only if it's allocated to Zone2)</p> <p>Reset type: SYSRSn</p>

**3.14.7.12 Z2\_EXEONLYRAMR Register (Offset = 20h) [reset = 0h]**

Z2\_EXEONLYRAMR is shown in [Figure 3-112](#) and described in [Table 3-120](#).

Return to the [Summary Table](#).

Zone 2 RAM Execute\_Only Block Register

**Figure 3-112. Z2\_EXEONLYRAMR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
EXEONLY_RA M7	EXEONLY_RA M6	EXEONLY_RA M5	EXEONLY_RA M4	EXEONLY_RA M3	EXEONLY_RA M2	EXEONLY_RA M1	EXEONLY_RA M0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-120. Z2\_EXEONLYRAMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	EXEONLY_RAM7	R	0h	Value in this field gets loaded from Z2OTP_EXEONLYRAM[7:7] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS7 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for LS7 RAM (only if it's allocated to Zone2)  Reset type: SYSRSn
6	EXEONLY_RAM6	R	0h	Value in this field gets loaded from Z2OTP_EXEONLYRAM[6:6] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS6 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for LS6 RAM (only if it's allocated to Zone2)  Reset type: SYSRSn
5	EXEONLY_RAM5	R	0h	Value in this field gets loaded from Z2OTP_EXEONLYRAM[5:5] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS5 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for LS5 RAM (only if it's allocated to Zone2)  Reset type: SYSRSn

**Table 3-120. Z2\_EXEONLYRAMR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	EXEONLY_RAM4	R	0h	Value in this field gets loaded from Z2OTP_EXEONLYRAM[4:4] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS4 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for LS4 RAM (only if it's allocated to Zone2) Reset type: SYSRSn
3	EXEONLY_RAM3	R	0h	Value in this field gets loaded from Z2OTP_EXEONLYRAM[3:3] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS3 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for LS3 RAM (only if it's allocated to Zone2) Reset type: SYSRSn
2	EXEONLY_RAM2	R	0h	Value in this field gets loaded from Z2OTP_EXEONLYRAM[2:2] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS2 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for LS2 RAM (only if it's allocated to Zone2) Reset type: SYSRSn
1	EXEONLY_RAM1	R	0h	Value in this field gets loaded from Z2OTP_EXEONLYRAM[1:1] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS1 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for LS1 RAM (only if it's allocated to Zone2) Reset type: SYSRSn
0	EXEONLY_RAM0	R	0h	Value in this field gets loaded from Z2OTP_EXEONLYRAM[0:0] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.  0 : Execute-Only protection is enabled for LS0 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for LS0 RAM (only if it's allocated to Zone2) Reset type: SYSRSn



### 3.14.8 DCSM\_BANK1\_Z1\_REGS Registers

Table 3-121 lists the DCSM\_BANK1\_Z1\_REGS registers. All register offset addresses not listed in Table 3-121 should be considered as reserved locations and the register contents should not be modified.

**Table 3-121. DCSM\_BANK1\_Z1\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	B1_Z1_LINKPOINTER	Zone 1 Link Pointer for flash BANK1		<a href="#">Go</a>
6h	B1_Z1_LINKPOINTERERR	Link Pointer Error for flash BANK1		<a href="#">Go</a>
1Ah	B1_Z1_GRABSECTR	Zone 1 Grab Flash BANK1 Sectors Register		<a href="#">Go</a>
1Eh	B1_Z1_EXEONLYSECTR	Zone 1 Flash BANK1 Execute_Only Sector Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-122 shows the codes that are used for access types in this section.

**Table 3-122. DCSM\_BANK1\_Z1\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.8.1 B1\_Z1\_LINKPOINTER Register (Offset = 0h) [reset = E000000h]

B1\_Z1\_LINKPOINTER is shown in [Figure 3-113](#) and described in [Table 3-123](#).

Return to the [Summary Table](#).

Zone 1 Link Pointer for flash BANK1

**Figure 3-113. B1\_Z1\_LINKPOINTER Register**

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

**Table 3-123. B1\_Z1\_LINKPOINTER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	7h	Reserved
28-0	LINKPOINTER	R	0h	This is resolved Link-Pointer for Zone1 zone select block USER OTP of Flash BANK1. This is generated by using three physical Link-Pointer values loaded from OTP in Flash BANK1. Reset type: SYSRSn

### 3.14.8.2 B1\_Z1\_LINKPOINTERERR Register (Offset = 6h) [reset = 0h]

B1\_Z1\_LINKPOINTERERR is shown in [Figure 3-114](#) and described in [Table 3-124](#).

Return to the [Summary Table](#).

Link Pointer Error for flash BANK1

**Figure 3-114. B1\_Z1\_LINKPOINTERERR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED			Z1_LINKPOINTERERR												
R-0-0h			R-0h												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z1_LINKPOINTERERR															
R-0h															

**Table 3-124. B1\_Z1\_LINKPOINTERERR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R-0	0h	Reserved
28-0	Z1_LINKPOINTERERR	R	0h	<p>These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded from USER OTP in Flash BANK1</p> <p>0 : No Error.</p> <p>Other : Error on bit positions which is set to 1.</p> <p>Reset type: SYSRSn</p>

### 3.14.8.3 B1\_Z1\_GRABSECTR Register (Offset = 1Ah) [reset = 0h]

B1\_Z1\_GRABSECTR is shown in [Figure 3-115](#) and described in [Table 3-125](#).

Return to the [Summary Table](#).

Zone 1 Grab Flash BANK1 Sectors Register

**Figure 3-115. B1\_Z1\_GRABSECTR Register**

31	30	29	28	27	26	25	24
GRAB_SECT15		GRAB_SECT14		GRAB_SECT13		GRAB_SECT12	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECT11		GRAB_SECT10		GRAB_SECT9		GRAB_SECT8	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECT7		GRAB_SECT6		GRAB_SECT5		GRAB_SECT4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_SECT3		GRAB_SECT2		GRAB_SECT1		GRAB_SECT0	
R-0h		R-0h		R-0h		R-0h	

**Table 3-125. B1\_Z1\_GRABSECTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT15	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[31:30] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 15 is inaccessible. 01 : Request to allocate Flash Sector 15 to Zone1. 10 : No request for Flash Sector 15 11 : No request for Flash Sector 15 when this zone is UNLOCKED. Else Flash Sector 15 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn
29-28	GRAB_SECT14	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[29:28] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 14 is inaccessible. 01 : Request to allocate Flash Sector 14 to Zone1. 10 : No request for Flash Sector 14 11 : No request for Flash Sector 14 when this zone is UNLOCKED. Else Flash Sector 14 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn
27-26	GRAB_SECT13	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[27:26] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 13 is inaccessible. 01 : Request to allocate Flash Sector 13 to Zone1. 10 : No request for Flash Sector 13 11 : No request for Flash Sector 13 when this zone is UNLOCKED. Else Flash Sector 13 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn

**Table 3-125. B1\_Z1\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT12	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[25:24] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 12 is inaccessible. 01 : Request to allocate Flash Sector 12 to Zone1. 10 : No request for Flash Sector 12 11 : No request for Flash Sector 12 when this zone is UNLOCKED. Else Flash Sector 12 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
23-22	GRAB_SECT11	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[23:22] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 11 is inaccessible. 01 : Request to allocate Flash Sector 11 to Zone1. 10 : No request for Flash Sector 11 11 : No request for Flash Sector 11 when this zone is UNLOCKED. Else Flash Sector 11 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
21-20	GRAB_SECT10	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[21:20] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 10 is inaccessible. 01 : Request to allocate Flash Sector 10 to Zone1. 10 : No request for Flash Sector 10 11 : No request for Flash Sector 10 when this zone is UNLOCKED. Else Flash Sector 10 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
19-18	GRAB_SECT9	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[19:18] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 9 is inaccessible. 01 : Request to allocate Flash Sector 9 to Zone1. 10 : No request for Flash Sector 9 11 : No request for Flash Sector 9 when this zone is UNLOCKED. Else Flash Sector 9 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
17-16	GRAB_SECT8	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[17:16] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 8 is inaccessible. 01 : Request to allocate Flash Sector 8 to Zone1. 10 : No request for Flash Sector 8 11 : No request for Flash sector 8 when this zone is UNLOCKED. Else Flash sector 8 is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 3-125. B1\_Z1\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15-14	GRAB_SECT7	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[15:14] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 7 is inaccessible. 01 : Request to allocate Flash Sector 7 to Zone1. 10 : No request for Flash Sector 7 11 : No request for Flash Sector 7 when this zone is UNLOCKED. Else Flash Sector 7 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
13-12	GRAB_SECT6	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[13:12] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 6 is inaccessible. 01 : Request to allocate Flash Sector 6 to Zone1. 10 : No request for Flash Sector 6 11 : No request for Flash Sector 6 when this zone is UNLOCKED. Else Flash Sector 6 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_SECT5	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[11:10] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 5 is inaccessible. 01 : Request to allocate Flash Sector 5 to Zone1. 10 : No request for Flash Sector 5 11 : No request for Flash Sector 5 when this zone is UNLOCKED. Else Flash Sector 5 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
9-8	GRAB_SECT4	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[9:8] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 4 is inaccessible. 01 : Request to allocate Flash Sector 4 to Zone1. 10 : No request for Flash Sector 4 11 : No request for Flash Sector 4 when this zone is UNLOCKED. Else Flash Sector 4 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	GRAB_SECT3	R	0h	Value in this field gets loaded from B1_Z1OTP_GRABSECTR[7:6] when a read is issued to address location of B1_Z1OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 3 is inaccessible. 01 : Request to allocate Flash Sector 3 to Zone1. 10 : No request for Flash Sector 3 11 : No request for Flash Sector 3 when this zone is UNLOCKED. Else Flash Sector 3 is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 3-125. B1\_Z1\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-4	GRAB_SECT2	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_GRABSECT[5:4] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.</p> <p>00 : Invalid. Flash Sector 2 is inaccessible.            01 : Request to allocate Flash Sector 2 to Zone1.            10 : No request for Flash Sector 2            11 : No request for Flash Sector 2 when this zone is UNLOCKED. Else Flash Sector 2 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
3-2	GRAB_SECT1	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_GRABSECT[3:2] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.</p> <p>00 : Invalid. Flash Sector 1 is inaccessible.            01 : Request to allocate Flash Sector 1 to Zone1.            10 : No request for Flash Sector 1            11 : No request for Flash sector 1 when this zone is UNLOCKED. Else Flash sector 1 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>
1-0	GRAB_SECT0	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_GRABSECT[1:0] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.</p> <p>00 : Invalid. Flash Sector 0 is inaccessible.            01 : Request to allocate Flash Sector 0 to Zone1.            10 : No request for Flash Sector 0            11 : No request for Flash Sector 0 when this zone is UNLOCKED. Else Flash Sector 0 is inaccessible if this zone is LOCKED.            Reset type: SYSRSn</p>

**3.14.8.4 B1\_Z1\_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]**

B1\_Z1\_EXEONLYSECTR is shown in [Figure 3-116](#) and described in [Table 3-126](#).

Return to the [Summary Table](#).

Zone 1 Flash BANK1 Execute\_Only Sector Register

**Figure 3-116. B1\_Z1\_EXEONLYSECTR 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
EXEONLY_SE CT15	EXEONLY_SE CT14	EXEONLY_SE CT13	EXEONLY_SE CT12	EXEONLY_SE CT11	EXEONLY_SE CT10	EXEONLY_SE CT9	EXEONLY_SE CT8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
EXEONLY_SE CT7	EXEONLY_SE CT6	EXEONLY_SE CT5	EXEONLY_SE CT4	EXEONLY_SE CT3	EXEONLY_SE CT2	EXEONLY_SE CT1	EXEONLY_SE CT0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-126. B1\_Z1\_EXEONLYSECTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	EXEONLY_SECT15	R	0h	Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[15:15] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 15 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 15 (only if it's allocated to Zone1)  Reset type: SYSRSn
14	EXEONLY_SECT14	R	0h	Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[14:14] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 14 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 14 (only if it's allocated to Zone1)  Reset type: SYSRSn
13	EXEONLY_SECT13	R	0h	Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[13:13] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 13 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 13 (only if it's allocated to Zone1)  Reset type: SYSRSn



**Table 3-126. B1\_Z1\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	EXEONLY_SECT12	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[12:12] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 12 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 12 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
11	EXEONLY_SECT11	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[11:11] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 11 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 11 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
10	EXEONLY_SECT10	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[10:10] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 10 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 10 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
9	EXEONLY_SECT9	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[9:9] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 9 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 9 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
8	EXEONLY_SECT8	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[8:8] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 8 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 8 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>
7	EXEONLY_SECT7	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[7:7] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 7 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 7 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>

**Table 3-126. B1\_Z1\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	EXEONLY_SECT6	R	0h	Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[6:6] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 6 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 6 (only if it's allocated to Zone1) Reset type: SYSRSn
5	EXEONLY_SECT5	R	0h	Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[5:5] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 5 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 5 (only if it's allocated to Zone1) Reset type: SYSRSn
4	EXEONLY_SECT4	R	0h	Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[4:4] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 4 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 4 (only if it's allocated to Zone1) Reset type: SYSRSn
3	EXEONLY_SECT3	R	0h	Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[3:3] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 3 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 3 (only if it's allocated to Zone1) Reset type: SYSRSn
2	EXEONLY_SECT2	R	0h	Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[2:2] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 2 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 2 (only if it's allocated to Zone1) Reset type: SYSRSn
1	EXEONLY_SECT1	R	0h	Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[1:1] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 1 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector 1 (only if it's allocated to Zone1) Reset type: SYSRSn

**Table 3-126. B1\_Z1\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	EXEONLY_SECT0	R	0h	<p>Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[0:0] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 0 (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 0 (only if it's allocated to Zone1)</p> <p>Reset type: SYSRSn</p>

### 3.14.9 DCSM\_BANK1\_Z2\_REGS Registers

Table 3-127 lists the DCSM\_BANK1\_Z2\_REGS registers. All register offset addresses not listed in Table 3-127 should be considered as reserved locations and the register contents should not be modified.

**Table 3-127. DCSM\_BANK1\_Z2\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	B1_Z2_LINKPOINTER	Zone 2 Link Pointer for flash BANK1		<a href="#">Go</a>
6h	B1_Z2_LINKPOINTERERR	Link Pointer Error for flash BANK1		<a href="#">Go</a>
1Ah	B1_Z2_GRABSECTR	Zone 2 Grab Flash BANK1 Sectors Register		<a href="#">Go</a>
1Eh	B1_Z2_EXEONLYSECTR	Zone 2 Flash BANK1 Execute_Only Sector Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-128 shows the codes that are used for access types in this section.

**Table 3-128. DCSM\_BANK1\_Z2\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

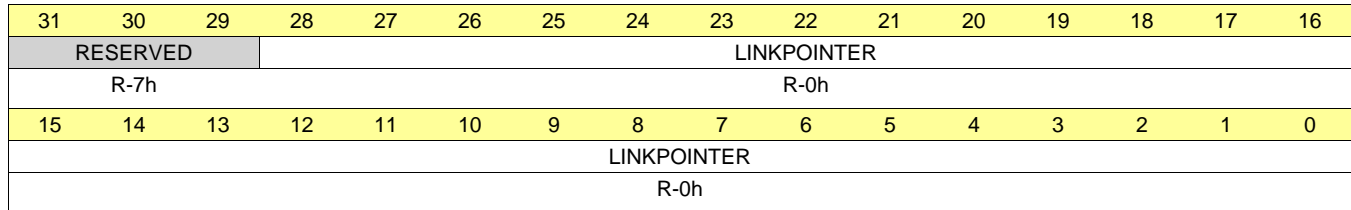
**3.14.9.1 B1\_Z2\_LINKPOINTER Register (Offset = 0h) [reset = E000000h]**

B1\_Z2\_LINKPOINTER is shown in [Figure 3-117](#) and described in [Table 3-129](#).

Return to the [Summary Table](#).

Zone 2 Link Pointer for flash BANK1

**Figure 3-117. B1\_Z2\_LINKPOINTER Register**



**Table 3-129. B1\_Z2\_LINKPOINTER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	7h	Reserved
28-0	LINKPOINTER	R	0h	This is resolved Link-Pointer for Zone2 zone select block USER OTP of Flash BANK1. This is generated by using three physical Link-Pointer values loaded from OTP in Flash BANK1. Reset type: SYSRSn

### 3.14.9.2 B1\_Z2\_LINKPOINTERERR Register (Offset = 6h) [reset = 0h]

B1\_Z2\_LINKPOINTERERR is shown in [Figure 3-118](#) and described in [Table 3-130](#).

Return to the [Summary Table](#).

Link Pointer Error for flash BANK1

**Figure 3-118. B1\_Z2\_LINKPOINTERERR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				Z2_LINKPOINTERERR											
R-0-0h				R-0h											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z2_LINKPOINTERERR															
R-0h															

**Table 3-130. B1\_Z2\_LINKPOINTERERR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R-0	0h	Reserved
28-0	Z2_LINKPOINTERERR	R	0h	These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded from USER OTP in Flash BANK1 0 : No Error. Other : Error on bit positions which is set to 1. Reset type: SYSRSn

### 3.14.9.3 B1\_Z2\_GRABSECTR Register (Offset = 1Ah) [reset = 0h]

B1\_Z2\_GRABSECTR is shown in [Figure 3-119](#) and described in [Table 3-131](#).

Return to the [Summary Table](#).

Zone 2 Grab Flash BANK1 Sectors Register

**Figure 3-119. B1\_Z2\_GRABSECTR Register**

31	30	29	28	27	26	25	24
GRAB_SECT15		GRAB_SECT14		GRAB_SECT13		GRAB_SECT12	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECT11		GRAB_SECT10		GRAB_SECT9		GRAB_SECT8	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECT7		GRAB_SECT6		GRAB_SECT5		GRAB_SECT4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_SECT3		GRAB_SECT2		GRAB_SECT1		GRAB_SECT0	
R-0h		R-0h		R-0h		R-0h	

**Table 3-131. B1\_Z2\_GRABSECTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT15	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECTR[31:30] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 15 is inaccessible. 01 : Request to allocate Flash Sector 15 to Zone2. 10 : No request for Flash Sector 15 11 : No request for Flash Sector 15 when this zone is UNLOCKED. Else Flash Sector 15 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn
29-28	GRAB_SECT14	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECTR[29:28] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 14 is inaccessible. 01 : Request to allocate Flash Sector 14 to Zone2. 10 : No request for Flash Sector 14 11 : No request for Flash Sector 14 when this zone is UNLOCKED. Else Flash Sector 14 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn
27-26	GRAB_SECT13	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECTR[27:26] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 13 is inaccessible. 01 : Request to allocate Flash Sector 13 to Zone2. 10 : No request for Flash Sector 13 11 : No request for Flash Sector 13 when this zone is UNLOCKED. Else Flash Sector 13 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn

**Table 3-131. B1\_Z2\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT12	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECTR[25:24] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 12 is inaccessible. 01 : Request to allocate Flash Sector 12 to Zone2. 10 : No request for Flash Sector 12 11 : No request for Flash Sector 12 when this zone is UNLOCKED. Else Flash Sector 12 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
23-22	GRAB_SECT11	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECTR[23:22] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 11 is inaccessible. 01 : Request to allocate Flash Sector 11 to Zone2. 10 : No request for Flash Sector 11 11 : No request for Flash Sector 11 when this zone is UNLOCKED. Else Flash Sector 11 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
21-20	GRAB_SECT10	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECTR[21:20] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 10 is inaccessible. 01 : Request to allocate Flash Sector 10 to Zone2. 10 : No request for Flash Sector 10 11 : No request for Flash Sector 10 when this zone is UNLOCKED. Else Flash Sector 10 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
19-18	GRAB_SECT9	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECTR[19:18] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 9 is inaccessible. 01 : Request to allocate Flash Sector 9 to Zone2. 10 : No request for Flash Sector 9 11 : No request for Flash Sector 9 when this zone is UNLOCKED. Else Flash Sector 9 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
17-16	GRAB_SECT8	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECTR[17:16] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 8 is inaccessible. 01 : Request to allocate Flash Sector 8 to Zone2. 10 : No request for Flash Sector 8 11 : No request for Flash sector 8 when this zone is UNLOCKED. Else Flash sector 8 is inaccessible if this zone is LOCKED. Reset type: SYSRSn



**Table 3-131. B1\_Z2\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15-14	GRAB_SECT7	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_GRABSECTR[15:14] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.</p> <p>00 : Invalid. Flash Sector 7 is inaccessible.  01 : Request to allocate Flash Sector 7 to Zone2.  10 : No request for Flash Sector 7  11 : No request for Flash Sector 7 when this zone is UNLOCKED. Else Flash Sector 7 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn</p>
13-12	GRAB_SECT6	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_GRABSECTR[13:12] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.</p> <p>00 : Invalid. Flash Sector 6 is inaccessible.  01 : Request to allocate Flash Sector 6 to Zone2.  10 : No request for Flash Sector 6  11 : No request for Flash Sector 6 when this zone is UNLOCKED. Else Flash Sector 6 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn</p>
11-10	GRAB_SECT5	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_GRABSECTR[11:10] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.</p> <p>00 : Invalid. Flash Sector 5 is inaccessible.  01 : Request to allocate Flash Sector 5 to Zone2.  10 : No request for Flash Sector 5  11 : No request for Flash Sector 5 when this zone is UNLOCKED. Else Flash Sector 5 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn</p>
9-8	GRAB_SECT4	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_GRABSECTR[9:8] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.</p> <p>00 : Invalid. Flash Sector 4 is inaccessible.  01 : Request to allocate Flash Sector 4 to Zone2.  10 : No request for Flash Sector 4  11 : No request for Flash Sector 4 when this zone is UNLOCKED. Else Flash Sector 4 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn</p>
7-6	GRAB_SECT3	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_GRABSECTR[7:6] when a read is issued to address location of B1_Z2OTP_GRABSECTR in USER OTP of Flash BANK1.</p> <p>00 : Invalid. Flash Sector 3 is inaccessible.  01 : Request to allocate Flash Sector 3 to Zone2.  10 : No request for Flash Sector 3  11 : No request for Flash Sector 3 when this zone is UNLOCKED. Else Flash Sector 3 is inaccessible if this zone is LOCKED.  Reset type: SYSRSn</p>

**Table 3-131. B1\_Z2\_GRABSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-4	GRAB_SECT2	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECT[5:4] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 2 is inaccessible. 01 : Request to allocate Flash Sector 2 to Zone2. 10 : No request for Flash Sector 2 11 : No request for Flash Sector 2 when this zone is UNLOCKED. Else Flash Sector 2 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
3-2	GRAB_SECT1	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECT[1:0] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 1 is inaccessible. 01 : Request to allocate Flash Sector 1 to Zone2. 10 : No request for Flash Sector 1 11 : No request for Flash sector 1 when this zone is UNLOCKED. Else Flash sector 1 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
1-0	GRAB_SECT0	R	0h	Value in this field gets loaded from B1_Z2OTP_GRABSECT[1:0] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.  00 : Invalid. Flash Sector 0 is inaccessible. 01 : Request to allocate Flash Sector 0 to Zone2. 10 : No request for Flash Sector 0 11 : No request for Flash Sector 0 when this zone is UNLOCKED. Else Flash Sector 0 is inaccessible if this zone is LOCKED. Reset type: SYSRSn

### 3.14.9.4 B1\_Z2\_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]

B1\_Z2\_EXEONLYSECTR is shown in [Figure 3-120](#) and described in [Table 3-132](#).

Return to the [Summary Table](#).

Zone 2 Flash BANK1 Execute\_Only Sector Register

**Figure 3-120. B1\_Z2\_EXEONLYSECTR 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
EXEONLY_SE CT15	EXEONLY_SE CT14	EXEONLY_SE CT13	EXEONLY_SE CT12	EXEONLY_SE CT11	EXEONLY_SE CT10	EXEONLY_SE CT9	EXEONLY_SE CT8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
EXEONLY_SE CT7	EXEONLY_SE CT6	EXEONLY_SE CT5	EXEONLY_SE CT4	EXEONLY_SE CT3	EXEONLY_SE CT2	EXEONLY_SE CT1	EXEONLY_SE CT0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-132. B1\_Z2\_EXEONLYSECTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	EXEONLY_SECT15	R	0h	Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[15:15] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 15 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 15 (only if it's allocated to Zone2)  Reset type: SYSRSn
14	EXEONLY_SECT14	R	0h	Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[14:14] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 14 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 14 (only if it's allocated to Zone2)  Reset type: SYSRSn
13	EXEONLY_SECT13	R	0h	Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[13:13] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 13 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 13 (only if it's allocated to Zone2)  Reset type: SYSRSn

**Table 3-132. B1\_Z2\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	EXEONLY_SECT12	R	0h	Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[12:12] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 12 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 12 (only if it's allocated to Zone2) Reset type: SYSRSn
11	EXEONLY_SECT11	R	0h	Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[11:11] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 11 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 11 (only if it's allocated to Zone2) Reset type: SYSRSn
10	EXEONLY_SECT10	R	0h	Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[10:10] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 10 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 10 (only if it's allocated to Zone2) Reset type: SYSRSn
9	EXEONLY_SECT9	R	0h	Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[9:9] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 9 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 9 (only if it's allocated to Zone2) Reset type: SYSRSn
8	EXEONLY_SECT8	R	0h	Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[8:8] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 8 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 8 (only if it's allocated to Zone2) Reset type: SYSRSn
7	EXEONLY_SECT7	R	0h	Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[7:7] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.  0 : Execute-Only protection is enabled for Flash Sector 7 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 7 (only if it's allocated to Zone2) Reset type: SYSRSn

**Table 3-132. B1\_Z2\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	EXEONLY_SECT6	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[6:6] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 6 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 6 (only if it's allocated to Zone2) Reset type: SYSRSn</p>
5	EXEONLY_SECT5	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[5:5] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 5 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 5 (only if it's allocated to Zone2) Reset type: SYSRSn</p>
4	EXEONLY_SECT4	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[4:4] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 4 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 4 (only if it's allocated to Zone2) Reset type: SYSRSn</p>
3	EXEONLY_SECT3	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[3:3] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 3 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 3 (only if it's allocated to Zone2) Reset type: SYSRSn</p>
2	EXEONLY_SECT2	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[2:2] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 2 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 2 (only if it's allocated to Zone2) Reset type: SYSRSn</p>
1	EXEONLY_SECT1	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[1:1] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 1 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector 1 (only if it's allocated to Zone2) Reset type: SYSRSn</p>

**Table 3-132. B1\_Z2\_EXEONLYSECTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	EXEONLY_SECT0	R	0h	<p>Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[0:0] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.</p> <p>0 : Execute-Only protection is enabled for Flash Sector 0 (only if it's allocated to Zone2)</p> <p>1 : Execute-Only protection is disabled for Flash Sector 0 (only if it's allocated to Zone2)</p> <p>Reset type: SYSRSn</p>

### 3.14.10 DCSM\_COMMON\_REGS Registers

Table 3-133 lists the DCSM\_COMMON\_REGS registers. All register offset addresses not listed in Table 3-133 should be considered as reserved locations and the register contents should not be modified.

**Table 3-133. DCSM\_COMMON\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	FLSEM	Flash Wrapper Semaphore Register	EALLOW	<a href="#">Go</a>
2h	B0_SECTSTAT	Flash BANK0 Sectors Status Register		<a href="#">Go</a>
4h	RAMSTAT	RAM Status Register		<a href="#">Go</a>
8h	B1_SECTSTAT	Flash BANK1 Sectors Status Register		<a href="#">Go</a>
Ah	SECERRSTAT	Security Error Status Register		<a href="#">Go</a>
Ch	SECERRCLR	Security Error Clear Register		<a href="#">Go</a>
Eh	SECERRFRC	Security Error Force Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-134 shows the codes that are used for access types in this section.

**Table 3-134. DCSM\_COMMON\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.10.1 FLSEM Register (Offset = 0h) [reset = 0h]**

FLSEM is shown in [Figure 3-121](#) and described in [Table 3-135](#).

Return to the [Summary Table](#).

Flash Wrapper Semaphore Register

**Figure 3-121. FLSEM 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
KEY								RESERVED						SEM	
R-0/W-0h								R-0h						R/W-0h	

**Table 3-135. FLSEM Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	KEY	R-0/W	0h	Writing a value 0xA5 into this field will allow the writing of the SEM bits, else writes are ignored. Reads will return 0. Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1-0	SEM	R/W	0h	00 : C28X Flash Wrapper registers can be written by code running from non-secure zone. 01 : Flash Wrapper registers can be written by code running from Zone1 security zone only. User must set this value to perform flash operation on flash sectors of Zone1. 10 : C28X Flash Wrapper registers can be written by code running from Zone2 security zone only. User must set this value to perform flash operation on flash sectors of Zone2. 11 : C28X Flash Wrapper registers can be written by code running from non-secure zone.  Allowed State Transitions in this field. 00 to 11 : Code running from anywhere. 11 to 00 : Not allowed. 00/11 to 01 : Code running from Zone1 only can perform this transition. 01 to 00/11 : Code running from Zone1 only can perform this transition. 00/11 to 10 : Code running from Zone2 only can perform this transition. 10 to 00/11 : Code running from Zone2 can perform this transition Reset type: SYSRSn



### 3.14.10.2 B0\_SECTSTAT Register (Offset = 2h) [reset = 0h]

B0\_SECTSTAT is shown in [Figure 3-122](#) and described in [Table 3-136](#).

Return to the [Summary Table](#).

Flash BANK0 Sectors Status Register

**Figure 3-122. B0\_SECTSTAT Register**

31	30	29	28	27	26	25	24
STATUS_SECT15		STATUS_SECT14		STATUS_SECT13		STATUS_SECT12	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
STATUS_SECT11		STATUS_SECT10		STATUS_SECT9		STATUS_SECT8	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
STATUS_SECT7		STATUS_SECT6		STATUS_SECT5		STATUS_SECT4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
STATUS_SECT3		STATUS_SECT2		STATUS_SECT1		STATUS_SECT0	
R-0h		R-0h		R-0h		R-0h	

**Table 3-136. B0\_SECTSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	STATUS_SECT15	R	0h	Reflects the status of Flash BANK0 Sector 15.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
29-28	STATUS_SECT14	R	0h	Reflects the status of Flash BANK0 Sector 14.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
27-26	STATUS_SECT13	R	0h	Reflects the status of Flash BANK0 Sector 13.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
25-24	STATUS_SECT12	R	0h	Reflects the status of Flash BANK0 Sector 12.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 3-136. B0\_SECTSTAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	STATUS_SECT11	R	0h	Reflects the status of Flash BANK0 Sector 11. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
21-20	STATUS_SECT10	R	0h	Reflects the status of Flash BANK0 Sector 10. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
19-18	STATUS_SECT9	R	0h	Reflects the status of Flash BANK0 Sector 9. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
17-16	STATUS_SECT8	R	0h	Reflects the status of Flash BANK0 sector 8. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
15-14	STATUS_SECT7	R	0h	Reflects the status of Flash BANK0 Sector 7. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
13-12	STATUS_SECT6	R	0h	Reflects the status of Flash BANK0 Sector 6. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
11-10	STATUS_SECT5	R	0h	Reflects the status of Flash BANK0 Sector 5. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 3-136. B0\_SECTSTAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	STATUS_SECT4	R	0h	Reflects the status of Flash BANK0 Sector 4.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
7-6	STATUS_SECT3	R	0h	Reflects the status of Flash BANK0 Sector 3.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
5-4	STATUS_SECT2	R	0h	Reflects the status of Flash BANK0 Sector 2.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
3-2	STATUS_SECT1	R	0h	Reflects the status of Flash BANK0 sector 1.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
1-0	STATUS_SECT0	R	0h	Reflects the status of Flash BANK0 Sector 0.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

### 3.14.10.3 RAMSTAT Register (Offset = 4h) [reset = 0h]

RAMSTAT is shown in [Figure 3-123](#) and described in [Table 3-137](#).

Return to the [Summary Table](#).

RAM Status Register

**Figure 3-123. RAMSTAT 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
STATUS_RAM7		STATUS_RAM6		STATUS_RAM5		STATUS_RAM4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
STATUS_RAM3		STATUS_RAM2		STATUS_RAM1		STATUS_RAM0	
R-0h		R-0h		R-0h		R-0h	

**Table 3-137. RAMSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	STATUS_RAM7	R	0h	Reflects the status of LS7 RAM.  00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
13-12	STATUS_RAM6	R	0h	Reflects the status of LS6 RAM.  00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
11-10	STATUS_RAM5	R	0h	Reflects the status of LS5 RAM.  00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
9-8	STATUS_RAM4	R	0h	Reflects the status of LS4 RAM.  00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 3-137. RAMSTAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	STATUS_RAM3	R	0h	Reflects the status of LS3 RAM.  00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
5-4	STATUS_RAM2	R	0h	Reflects the status of LS2 RAM.  00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
3-2	STATUS_RAM1	R	0h	Reflects the status of LS1 RAM.  00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
1-0	STATUS_RAM0	R	0h	Reflects the status of LS0 RAM.  00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**3.14.10.4 B1\_SECTSTAT Register (Offset = 8h) [reset = 0h]**

B1\_SECTSTAT is shown in [Figure 3-124](#) and described in [Table 3-138](#).

Return to the [Summary Table](#).

Flash BANK1 Sectors Status Register

**Figure 3-124. B1\_SECTSTAT Register**

31	30	29	28	27	26	25	24
STATUS_SECT15		STATUS_SECT14		STATUS_SECT13		STATUS_SECT12	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
STATUS_SECT11		STATUS_SECT10		STATUS_SECT9		STATUS_SECT8	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
STATUS_SECT7		STATUS_SECT6		STATUS_SECT5		STATUS_SECT4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
STATUS_SECT3		STATUS_SECT2		STATUS_SECT1		STATUS_SECT0	
R-0h		R-0h		R-0h		R-0h	

**Table 3-138. B1\_SECTSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	STATUS_SECT15	R	0h	Reflects the status of Flash BANK1 Sector 15.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
29-28	STATUS_SECT14	R	0h	Reflects the status of Flash BANK1 Sector 14.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
27-26	STATUS_SECT13	R	0h	Reflects the status of Flash BANK1 Sector 13.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
25-24	STATUS_SECT12	R	0h	Reflects the status of Flash BANK1 Sector 12.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 3-138. B1\_SECTSTAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	STATUS_SECT11	R	0h	<p>Reflects the status of Flash BANK1 Sector 11.</p> <p>00 : Sector is in-accessible  01 : Sector belongs to Zone1.  10 : Sector belongs to Zone2.  11: Sector is un-secure and code running in both zone have full access to it.  Reset type: SYSRSn</p>
21-20	STATUS_SECT10	R	0h	<p>Reflects the status of Flash BANK1 Sector 10.</p> <p>00 : Sector is in-accessible  01 : Sector belongs to Zone1.  10 : Sector belongs to Zone2.  11: Sector is un-secure and code running in both zone have full access to it.  Reset type: SYSRSn</p>
19-18	STATUS_SECT9	R	0h	<p>Reflects the status of Flash BANK1 Sector 9.</p> <p>00 : Sector is in-accessible  01 : Sector belongs to Zone1.  10 : Sector belongs to Zone2.  11: Sector is un-secure and code running in both zone have full access to it.  Reset type: SYSRSn</p>
17-16	STATUS_SECT8	R	0h	<p>Reflects the status of Flash BANK1 sector 8.</p> <p>00 : Sector is in-accessible  01 : Sector belongs to Zone1.  10 : Sector belongs to Zone2.  11: Sector is un-secure and code running in both zone have full access to it.  Reset type: SYSRSn</p>
15-14	STATUS_SECT7	R	0h	<p>Reflects the status of Flash BANK1 Sector 7.</p> <p>00 : Sector is in-accessible  01 : Sector belongs to Zone1.  10 : Sector belongs to Zone2.  11: Sector is un-secure and code running in both zone have full access to it.  Reset type: SYSRSn</p>
13-12	STATUS_SECT6	R	0h	<p>Reflects the status of Flash BANK1 Sector 6.</p> <p>00 : Sector is in-accessible  01 : Sector belongs to Zone1.  10 : Sector belongs to Zone2.  11: Sector is un-secure and code running in both zone have full access to it.  Reset type: SYSRSn</p>
11-10	STATUS_SECT5	R	0h	<p>Reflects the status of Flash BANK1 Sector 5.</p> <p>00 : Sector is in-accessible  01 : Sector belongs to Zone1.  10 : Sector belongs to Zone2.  11: Sector is un-secure and code running in both zone have full access to it.  Reset type: SYSRSn</p>

**Table 3-138. B1\_SECTSTAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	STATUS_SECT4	R	0h	Reflects the status of Flash BANK1 Sector 4.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
7-6	STATUS_SECT3	R	0h	Reflects the status of Flash BANK1 Sector 3.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
5-4	STATUS_SECT2	R	0h	Reflects the status of Flash BANK1 Sector 2.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
3-2	STATUS_SECT1	R	0h	Reflects the status of Flash BANK1 sector 1.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
1-0	STATUS_SECT0	R	0h	Reflects the status of Flash BANK1 Sector 0.  00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn



**3.14.10.5 SECERRSTAT Register (Offset = Ah) [reset = 0h]**

SECERRSTAT is shown in [Figure 3-125](#) and described in [Table 3-139](#).

Return to the [Summary Table](#).

Security Error Status Register

**Figure 3-125. SECERRSTAT Register**

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

**Table 3-139. SECERRSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	ERR	R	0h	<p>This bit indicates if any error has occurred in the load of any security configuration from USER-OTP.</p> <p>0: No error has occurred in the load of security information from USER-OTP</p> <p>1: Error has occurred in the load of security information from USER-OTP</p> <p>Reset type: PORESETn</p>

**3.14.10.6 SECERRCLR Register (Offset = Ch) [reset = 0h]**

SECERRCLR is shown in [Figure 3-126](#) and described in [Table 3-140](#).

Return to the [Summary Table](#).

Security Error Clear Register

**Figure 3-126. SECERRCLR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															ERR
R-0-0h															R- 0/W1S -0h

**Table 3-140. SECERRCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	ERR	R-0/W1S	0h	A write of '1' clears the SECERRSTST.ERR bit. Write of '0' is ignored. This bit always reads back '0'. Reset type: N/A

**3.14.10.7 SECERRFRC Register (Offset = Eh) [reset = 0h]**

SECERRFRC is shown in [Figure 3-127](#) and described in [Table 3-141](#).

Return to the [Summary Table](#).

Security Error Force Register

**Figure 3-127. SECERRFRC Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															ERR
R-0-0h															R-0/W1S-0h

**Table 3-141. SECERRFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	ERR	R-0/W1S	0h	A write of '1' sets the SECERRSTST.ERR bit. Write of '0' is ignored. This bit always reads back '0'. Reset type: N/A

### 3.14.11 DEV\_CFG\_REGS Registers

Table 3-142 lists the DEV\_CFG\_REGS registers. All register offset addresses not listed in Table 3-142 should be considered as reserved locations and the register contents should not be modified.

**Table 3-142. DEV\_CFG\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
8h	PARTIDL	Lower 32-bit of Device PART Identification Number		<a href="#">Go</a>
Ah	PARTIDH	Upper 32-bit of Device PART Identification Number		<a href="#">Go</a>
Ch	REVID	Device Revision Number		<a href="#">Go</a>
3Ah	DC21	Device Capability: CLB		<a href="#">Go</a>
74h	FUSEERR	e-Fuse error Status register		<a href="#">Go</a>
82h	SOFTPRES0	Processing Block Software Reset register	EALLOW	<a href="#">Go</a>
86h	SOFTPRES2	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
88h	SOFTPRES3	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
8Ah	SOFTPRES4	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
8Eh	SOFTPRES6	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
90h	SOFTPRES7	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
92h	SOFTPRES8	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
94h	SOFTPRES9	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
96h	SOFTPRES10	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
9Ch	SOFTPRES13	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
9Eh	SOFTPRES14	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
A0h	SOFTPRES15	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
A2h	SOFTPRES16	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
A4h	SOFTPRES17	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
A6h	SOFTPRES18	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
A8h	SOFTPRES19	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
AAh	SOFTPRES20	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
ACh	SOFTPRES21	Peripheral Software Reset register	EALLOW	<a href="#">Go</a>
130h	TAP_STATUS	Status of JTAG State machine & Debugger Connect		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-143 shows the codes that are used for access types in this section.

**Table 3-143. DEV\_CFG\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
WOnce	WOnce	Write Write once
WSonce	WSonce	Write Set once
<b>Reset or Default Value</b>		

**Table 3-143. DEV\_CFG\_REGS Access Type Codes (continued)**

Access Type	Code	Description
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.11.1 PARTIDL Register (Offset = 8h) [reset = 0h]**

PARTIDL is shown in [Figure 3-128](#) and described in [Table 3-144](#).

Return to the [Summary Table](#).

Lower 32-bit of Device PART Identification Number

**Figure 3-128. PARTIDL Register**

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

**Table 3-144. PARTIDL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-24	RESERVED	R	0h	Reserved
23-16	FLASH_SIZE	R	0h	6 = 256KB 5 = 128KB Reset type: XRSn
15	RESERVED	R	0h	Reserved
14-13	INSTASPIN	R	0h	0 = InstaSPIN-MOTION + InstaSPIN-FOC 1 = InstaSPIN-FOC 2 = NONE 3 = NONE Reset type: XRSn
12	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10-8	PIN_COUNT	R	0h	0 = 56 pin 1 = 64 pin (Q100) 2 = 64 pin 3 = Reserved 4 = Reserved 5 = 100 pin 6 = Reserved 7 = Reserved Reset type: XRSn
7-6	QUAL	R	0h	0 = Engineering sample (TMX) 1 = Pilot production (TMP) 2 = Fully qualified (TMS) Reset type: XRSn
5	RESERVED	R	0h	Reserved

**Table 3-144. PARTIDL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-3	RESERVED	R	0h	Reserved
2-0	RESERVED	R	0h	Reserved

### 3.14.11.2 PARTIDH Register (Offset = Ah) [reset = 01000000h]

PARTIDH is shown in [Figure 3-129](#) and described in [Table 3-145](#).

Return to the [Summary Table](#).

Upper 32-bit of Device PART Identification Number

**Figure 3-129. PARTIDH Register**

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

**Table 3-145. PARTIDH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	DEVICE_CLASS_ID	R	1h	Device class ID Reset type: XRSn
23-16	PARTNO	R	0h	Refer to Datasheet for Device Part Number Reset type: XRSn
15-8	FAMILY	R	0h	Device Family 0x5 - SINGLE CORE Other values Reserved Reset type: XRSn
7-4	RESERVED	R	0h	Reserved
3-0	RESERVED	R	0h	Reserved



### 3.14.11.3 REVID Register (Offset = Ch) [reset = 0h]

REVID is shown in [Figure 3-130](#) and described in [Table 3-146](#).

Return to the [Summary Table](#).

Device Revision Number

**Figure 3-130. REVID 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																REVID															
R-0-0h																R-0h															

**Table 3-146. REVID Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-0	REVID	R	0h	Device Revision Reset type: N/A

**3.14.11.4 DC21 Register (Offset = 3Ah) [reset = 0h]**

DC21 is shown in [Figure 3-131](#) and described in [Table 3-147](#).

Return to the [Summary Table](#).

Device Capability: CLB

**Figure 3-131. DC21 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				CLB4	CLB3	CLB2	CLB1
R-0h				R-0h	R-0h	R-0h	R-0h

**Table 3-147. DC21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	CLB4	R	0h	CLB4 : 0: Feature not present on the device 1: Feature present on the device Reset type: XRSn
2	CLB3	R	0h	CLB3 : 0: Feature not present on the device 1: Feature present on the device Reset type: XRSn
1	CLB2	R	0h	CLB2 : 0: Feature not present on the device 1: Feature present on the device Reset type: XRSn
0	CLB1	R	0h	CLB1 : 0: Feature not present on the device 1: Feature present on the device Reset type: XRSn

### 3.14.11.5 FUSEERR Register (Offset = 74h) [reset = 0h]

FUSEERR is shown in [Figure 3-132](#) and described in [Table 3-148](#).

Return to the [Summary Table](#).

e-Fuse error Status register

**Figure 3-132. FUSEERR Register**

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

**Table 3-148. FUSEERR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5	ERR	R	0h	Efuse Self Test Error Status set by hardware after fuse self test completes, in case of self test error  0: No error during fuse self test 1: Fuse self test error Reset type: XRSn
4-0	ALERR	R	0h	Efuse Autoload Error Status set by hardware after fuse auto load completes  00000: No error in auto load Other: Non zero value indicates error in autoload  Note: [1] 10101 means a single-bit error during autoload. Since this gets corrected by the ECC mechanism, this value shouldn't be treated as an error condition. Reset type: XRSn

### 3.14.11.6 SOFTPRES0 Register (Offset = 82h) [reset = 0h]

SOFTPRES0 is shown in [Figure 3-133](#) and described in [Table 3-149](#).

Return to the [Summary Table](#).

Processing Block Software Reset register

When bits in this register are set, the respective module is in reset. All design data is lost and the module registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-133. SOFTPRES0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED	RESERVED	RESERVED	CPU1_CLA1
R-0-0h							R/W-0h

**Table 3-149. SOFTPRES0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	CPU1_CLA1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.7 SOFTPRES2 Register (Offset = 86h) [reset = 0h]**

SOFTPRES2 is shown in [Figure 3-134](#) and described in [Table 3-150](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-134. SOFTPRES2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
7	6	5	4	3	2	1	0
EPWM8	EPWM7	EPWM6	EPWM5	EPWM4	EPWM3	EPWM2	EPWM1
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 3-150. SOFTPRES2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	EPWM8	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
6	EPWM7	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
5	EPWM6	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
4	EPWM5	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
3	EPWM4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**Table 3-150. SOFTPRES2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	EPWM3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
1	EPWM2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	EPWM1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.14.11.8 SOFTPRES3 Register (Offset = 88h) [reset = 0h]

SOFTPRES3 is shown in [Figure 3-135](#) and described in [Table 3-151](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-135. SOFTPRES3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	ECAP7	ECAP6	ECAP5	ECAP4	ECAP3	ECAP2	ECAP1
	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-151. SOFTPRES3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	ECAP7	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
5	ECAP6	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
4	ECAP5	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
3	ECAP4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
2	ECAP3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
1	ECAP2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	ECAP1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.14.11.9 SOFTPRES4 Register (Offset = 8Ah) [reset = 0h]

SOFTPRES4 is shown in [Figure 3-136](#) and described in [Table 3-152](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-136. SOFTPRES4 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED	RESERVED	EQEP2	EQEP1
R-0-0h						R/W-0h	R/W-0h

**Table 3-152. SOFTPRES4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	EQEP2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	EQEP1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn



**3.14.11.10 SOFTPRES6 Register (Offset = 8Eh) [reset = 0h]**

SOFTPRES6 is shown in [Figure 3-137](#) and described in [Table 3-153](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-137. SOFTPRES6 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								RESE RVED	RESE RVED	RESE RVED	RESE RVED	RESE RVED	RESE RVED	RESE RVED	SD1
R-0-0h															R/W- 0h

**Table 3-153. SOFTPRES6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	SD1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.11 SOFTPRES7 Register (Offset = 90h) [reset = 0h]**

SOFTPRES7 is shown in [Figure 3-138](#) and described in [Table 3-154](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-138. SOFTPRES7 Register**

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

**Table 3-154. SOFTPRES7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	SCI_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	SCI_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.12 SOFTPRES8 Register (Offset = 92h) [reset = 0h]**

SOFTPRES8 is shown in [Figure 3-139](#) and described in [Table 3-155](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-139. SOFTPRES8 Register**

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

**Table 3-155. SOFTPRES8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-18	RESERVED	R-0	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	SPI_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	SPI_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.13 SOFTPRES9 Register (Offset = 94h) [reset = 0h]**

SOFTPRES9 is shown in [Figure 3-140](#) and described in [Table 3-156](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-140. SOFTPRES9 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						RESERVED	I2C_A
R-0-0h							R/W-0h

**Table 3-156. SOFTPRES9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	I2C_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.14 SOFTPRES10 Register (Offset = 96h) [reset = 0h]**

SOFTPRES10 is shown in [Figure 3-141](#) and described in [Table 3-157](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-141. SOFTPRES10 Register**

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

**Table 3-157. SOFTPRES10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	CAN_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	CAN_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.15 SOFTPRES13 Register (Offset = 9Ch) [reset = 0h]**

SOFTPRES13 is shown in [Figure 3-142](#) and described in [Table 3-158](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-142. SOFTPRES13 Register**

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

**Table 3-158. SOFTPRES13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	ADC_C	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
1	ADC_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	ADC_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.16 SOFTPRES14 Register (Offset = 9Eh) [reset = 0h]**

SOFTPRES14 is shown in [Figure 3-143](#) and described in [Table 3-159](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-143. SOFTPRES14 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	CMPSS7	CMPSS6	CMPSS5	CMPSS4	CMPSS3	CMPSS2	CMPSS1
	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-159. SOFTPRES14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	CMPSS7	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
5	CMPSS6	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
4	CMPSS5	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
3	CMPSS4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
2	CMPSS3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
1	CMPSS2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	CMPSS1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.17 SOFTPRES15 Register (Offset = A0h) [reset = 0h]**

SOFTPRES15 is shown in [Figure 3-144](#) and described in [Table 3-160](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-144. SOFTPRES15 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	PGA7	PGA6	PGA5	PGA4	PGA3	PGA2	PGA1
	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-160. SOFTPRES15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	PGA7	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
5	PGA6	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
4	PGA5	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
3	PGA4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
2	PGA3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
1	PGA2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	PGA1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn



**3.14.11.18 SOFTPRES16 Register (Offset = A2h) [reset = 0h]**

SOFTPRES16 is shown in [Figure 3-145](#) and described in [Table 3-161](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-145. SOFTPRES16 Register**

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

**Table 3-161. SOFTPRES16 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R-0	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	DAC_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
16	DAC_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
15-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

**3.14.11.19 SOFTPRES17 Register (Offset = A4h) [reset = 0h]**

SOFTPRES17 is shown in [Figure 3-146](#) and described in [Table 3-162](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-146. SOFTPRES17 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				CLB4	CLB3	CLB2	CLB1
R-0h				R/W-0h	R/W-0h	R-0h	R-0h

**Table 3-162. SOFTPRES17 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	CLB4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
2	CLB3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
1	CLB2	R	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	CLB1	R	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.20 SOFTPRES18 Register (Offset = A6h) [reset = 0h]**

SOFTPRES18 is shown in [Figure 3-147](#) and described in [Table 3-163](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-147. SOFTPRES18 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				RESERVED	RESERVED	RESERVED	RESERVED
R-0h							

**Table 3-163. SOFTPRES18 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

**3.14.11.21 SOFTPRES19 Register (Offset = A8h) [reset = 0h]**

SOFTPRES19 is shown in [Figure 3-148](#) and described in [Table 3-164](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-148. SOFTPRES19 Register**

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

**Table 3-164. SOFTPRES19 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	LIN_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**3.14.11.22 SOFTPRES20 Register (Offset = AAh) [reset = 0h]**

SOFTPRES20 is shown in [Figure 3-149](#) and described in [Table 3-165](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-149. SOFTPRES20 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						RESERVED	PMBUS_A
R-0h							R-0h

**Table 3-165. SOFTPRES20 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	PMBUS_A	R	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.14.11.23 SOFTPRES21 Register (Offset = ACh) [reset = 0h]

SOFTPRES21 is shown in [Figure 3-150](#) and described in [Table 3-166](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-150. SOFTPRES21 Register**

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

**Table 3-166. SOFTPRES21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

**3.14.11.24 TAP\_STATUS Register (Offset = 130h) [reset = 0h]**

TAP\_STATUS is shown in [Figure 3-151](#) and described in [Table 3-167](#).

Return to the [Summary Table](#).

Status of JTAG State machine &amp; Debugger Connect

**Figure 3-151. TAP\_STATUS Register**

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

**Table 3-167. TAP\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	DCON	R	0h	DebugConnect indication from IcePick. Reset type: POR
30-16	RESERVED	R-0	0h	Reserved
15-0	TAP_STATE	R	0h	TAP State Vector. With bits representing, Connect coresponding POTAP* output to the 0:TLR, 1:IDLE, 2:SELECTDR, 3:CAPDR, 4:SHIFTDTR, 5:EXIT1DR, 6:PAUSEDR, 7:EXIT2DR, 8:UPDTR, 9:SLECTIR, 10:CAPIR, 11:SHIFDIR, 12:EXIT1IR, 13:PAUSEIR, 14:EXIT2IR, 15:UPDIR, Reset type: POR

### 3.14.12 DMA\_CLA\_SRC\_SEL\_REGS Registers

Table 3-168 lists the DMA\_CLA\_SRC\_SEL\_REGS registers. All register offset addresses not listed in Table 3-168 should be considered as reserved locations and the register contents should not be modified.

**Table 3-168. DMA\_CLA\_SRC\_SEL\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	CLA1TASKSRCSELLOCK	CLA1 Task Trigger Source Select Lock Register	EALLOW	<a href="#">Go</a>
4h	DMACHSRCSELLOCK	DMA Channel Trigger Source Select Lock Register	EALLOW	<a href="#">Go</a>
6h	CLA1TASKSRCSEL1	CLA1 Task Trigger Source Select Register-1	EALLOW	<a href="#">Go</a>
8h	CLA1TASKSRCSEL2	CLA1 Task Trigger Source Select Register-2	EALLOW	<a href="#">Go</a>
16h	DMACHSRCSEL1	DMA Channel Trigger Source Select Register-1	EALLOW	<a href="#">Go</a>
18h	DMACHSRCSEL2	DMA Channel Trigger Source Select Register-2	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-169 shows the codes that are used for access types in this section.

**Table 3-169. DMA\_CLA\_SRC\_SEL\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
WOnce	WOnce	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.



**3.14.12.1 CLA1TASKSRCSELLOCK Register (Offset = 0h) [reset = 0h]**

CLA1TASKSRCSELLOCK is shown in [Figure 3-152](#) and described in [Table 3-170](#).

Return to the [Summary Table](#).

CLA1 Task Trigger Source Select Lock Register

**Figure 3-152. CLA1TASKSRCSELLOCK Register**

31	30	29	28	27	26	25	24		
RESERVED									
R-0-0h									
23	22	21	20	19	18	17	16		
RESERVED									
R-0-0h									
15	14	13	12	11	10	9	8		
RESERVED									
R-0-0h									
7	6	5	4	3	2	1	0		
RESERVED							CLA1TASKSR CSEL2	CLA1TASKSR CSEL1	
R-0-0h							R/WOnce-0h	R/WOnce-0h	

**Table 3-170. CLA1TASKSRCSELLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	CLA1TASKSRCSEL2	R/WOnce	0h	CLA1TASKSRCSEL2 Register Lock bit: 0: Respective register is not locked 1: Respective register is locked. Notes: [1] Any SOnce bit in this register, once set can only be cleared through a SYSRSn. Write of 0 to any bit of this register has no effect [2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed Reset type: SYSRSn
0	CLA1TASKSRCSEL1	R/WOnce	0h	CLA1TASKSRCSEL1 Register Lock bit: 0: Respective register is not locked 1: Respective register is locked. Notes: [1] Any SOnce bit in this register, once set can only be cleared through a SYSRSn. Write of 0 to any bit of this register has no effect [2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed Reset type: SYSRSn

**3.14.12.2 DMACHSRCSELLOCK Register (Offset = 4h) [reset = 0h]**

DMACHSRCSELLOCK is shown in [Figure 3-153](#) and described in [Table 3-171](#).

Return to the [Summary Table](#).

DMA Channel Trigger Source Select Lock Register

**Figure 3-153. DMACHSRCSELLOCK Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						DMACHSRCSE L2	DMACHSRCSE L1
R-0-0h						R/WSONce-0h	R/WSONce-0h

**Table 3-171. DMACHSRCSELLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	DMACHSRCSEL2	R/WSONce	0h	DMACHSRCSEL2 Register Lock bit: 0: Respective register is not locked 1: Respective register is locked. Notes: [1] Any SOnce bit in this register, once set can only be cleared through a SYSRSn. Write of 0 to any bit of this register has no effect [2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed Reset type: SYSRSn
0	DMACHSRCSEL1	R/WSONce	0h	DMACHSRCSEL1 Register Lock bit: 0: Respective register is not locked 1: Respective register is locked. Notes: [1] Any SOnce bit in this register, once set can only be cleared through a SYSRSn. Write of 0 to any bit of this register has no effect [2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed Reset type: SYSRSn

**3.14.12.3 CLA1TASKSRCSEL1 Register (Offset = 6h) [reset = 0h]**

CLA1TASKSRCSEL1 is shown in [Figure 3-154](#) and described in [Table 3-172](#).

Return to the [Summary Table](#).

CLA1 Task Trigger Source Select Register-1

**Figure 3-154. CLA1TASKSRCSEL1 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
TASK4								TASK3								TASK2								TASK1							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 3-172. CLA1TASKSRCSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	TASK4	R/W	0h	Selects the Trigger Source for TASK4 of CLA1 Reset type: SYSRSn
23-16	TASK3	R/W	0h	Selects the Trigger Source for TASK3 of CLA1 Reset type: SYSRSn
15-8	TASK2	R/W	0h	Selects the Trigger Source for TASK2 of CLA1 Reset type: SYSRSn
7-0	TASK1	R/W	0h	Selects the Trigger Source for TASK1 of CLA1 Reset type: SYSRSn

### 3.14.12.4 CLA1TASKSRCSEL2 Register (Offset = 8h) [reset = 0h]

CLA1TASKSRCSEL2 is shown in [Figure 3-155](#) and described in [Table 3-173](#).

Return to the [Summary Table](#).

CLA1 Task Trigger Source Select Register-2

**Figure 3-155. CLA1TASKSRCSEL2 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
TASK8								TASK7								TASK6								TASK5							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 3-173. CLA1TASKSRCSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	TASK8	R/W	0h	Selects the Trigger Source for TASK8 of CLA1 Reset type: SYSRSn
23-16	TASK7	R/W	0h	Selects the Trigger Source for TASK7 of CLA1 Reset type: SYSRSn
15-8	TASK6	R/W	0h	Selects the Trigger Source for TASK6 of CLA1 Reset type: SYSRSn
7-0	TASK5	R/W	0h	Selects the Trigger Source for TASK5 of CLA1 Reset type: SYSRSn

**3.14.12.5 DMACHSRCSEL1 Register (Offset = 16h) [reset = 0h]**

DMACHSRCSEL1 is shown in [Figure 3-156](#) and described in [Table 3-174](#).

Return to the [Summary Table](#).

DMA Channel Trigger Source Select Register-1

**Figure 3-156. DMACHSRCSEL1 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
CH4								CH3								CH2								CH1							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 3-174. DMACHSRCSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	CH4	R/W	0h	Selects the Trigger and Sync Source CH4 of DMA Reset type: SYSRSn
23-16	CH3	R/W	0h	Selects the Trigger and Sync Source CH3 of DMA Reset type: SYSRSn
15-8	CH2	R/W	0h	Selects the Trigger and Sync Source CH2 of DMA Reset type: SYSRSn
7-0	CH1	R/W	0h	Selects the Trigger and Sync Source CH1 of DMA Reset type: SYSRSn

### 3.14.12.6 DMACHSRCSEL2 Register (Offset = 18h) [reset = 0h]

DMACHSRCSEL2 is shown in [Figure 3-157](#) and described in [Table 3-175](#).

Return to the [Summary Table](#).

DMA Channel Trigger Source Select Register-2

**Figure 3-157. DMACHSRCSEL2 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																CH6						CH5									
R-0-0h																R/W-0h						R/W-0h									

**Table 3-175. DMACHSRCSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	CH6	R/W	0h	Selects the Trigger and Sync Source CH6 of DMA Reset type: SYSRSn
7-0	CH5	R/W	0h	Selects the Trigger and Sync Source CH5 of DMA Reset type: SYSRSn

### 3.14.13 FLASH\_CTRL\_REGS Registers

Table 3-176 lists the FLASH\_CTRL\_REGS registers. All register offset addresses not listed in Table 3-176 should be considered as reserved locations and the register contents should not be modified.

**Table 3-176. FLASH\_CTRL\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	FRDCNTL	Flash Read Control Register	EALLOW	<a href="#">Go</a>
1Eh	FBAC	Flash Bank Access Control Register	EALLOW	<a href="#">Go</a>
20h	FBFALLBACK	Flash Bank Fallback Power Register	EALLOW	<a href="#">Go</a>
22h	FBPRDY	Flash Bank Pump Ready Register	EALLOW	<a href="#">Go</a>
24h	FPAC1	Flash Pump Access Control Register 1	EALLOW	<a href="#">Go</a>
26h	FPAC2	Flash Pump Access Control Register 2	EALLOW	<a href="#">Go</a>
2Ah	FMSTAT	Flash Module Status Register	EALLOW	<a href="#">Go</a>
180h	FRD_INTF_CTRL	Flash Read Interface Control Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-177 shows the codes that are used for access types in this section.

**Table 3-177. FLASH\_CTRL\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.13.1 FRDCNTL Register (Offset = 0h) [reset = F00h]**

FRDCNTL is shown in [Figure 3-158](#) and described in [Table 3-178](#).

Return to the [Summary Table](#).

Flash Read Control Register

**Figure 3-158. FRDCNTL 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				RWAIT				RESERVED							
R-0h				R/W-Fh				R-0h							

**Table 3-178. FRDCNTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
11-8	RWAIT	R/W	Fh	Random read waitstate  These bits indicate how many waitstates are added to a flash read/fetch access. The RWAIT value can be set anywhere from 0 to 0xF. For a flash access, data is returned in RWAIT+1 SYSCLK cycles.  Note: The required wait states for each SYSCLK frequency can be found in the device data manual.  Reset type: SYSRSn
7-0	RESERVED	R	0h	Reserved



**3.14.13.2 FBAC Register (Offset = 1Eh) [reset = Fh]**

FBAC is shown in [Figure 3-159](#) and described in [Table 3-179](#).

Return to the [Summary Table](#).

Flash Bank Access Control Register

**Figure 3-159. FBAC 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																BAGP						RESERVED									
R-0h																R/W-0h															

**Table 3-179. FBAC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	BAGP	R/W	0h	Bank Active Grace Period. These bits contain the starting count value for the BAGP down counter. Any access to a given bank causes its BAGP counter to reload the BAGP value for that bank. After the last access to this flash bank, the down counter delays from 0 to 255 prescaled SYSCLK clock cycles before putting the bank into one of the fallback power modes as determined by the FBFALLBACK register. This value must be greater than 1 when the fallback mode is not ACTIVE.  Note: The prescaled clock used for the BAGP down counter is a clock divided by 16 from input SYSCLK. Reset type: SYRSn
7-0	RESERVED	R/W	Fh	Reserved

**3.14.13.3 FBFALLBACK Register (Offset = 20h) [reset = 0h]**

FBFALLBACK is shown in [Figure 3-160](#) and described in [Table 3-180](#).

Return to the [Summary Table](#).

Flash Bank Fallback Power Register

**Figure 3-160. FBFALLBACK 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				BNKPWR1		BNKPWR0	
R-0h				R/W-0h		R/W-0h	

**Table 3-180. FBFALLBACK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3-2	BNKPWR1	R/W	0h	Fall Back power mode 00 Sleep (Sense amplifiers and sense reference disabled) 01 Standby (Sense amplifiers disabled, but sense reference enabled) 10 Reserved 11 Active (Both sense amplifiers and sense reference enabled) Reset type: SYSRSn
1-0	BNKPWR0	R/W	0h	Fall Back power mode 00 Sleep (Sense amplifiers and sense reference disabled) 01 Standby (Sense amplifiers disabled, but sense reference enabled) 10 Reserved 11 Active (Both sense amplifiers and sense reference enabled) Reset type: SYSRSn

**3.14.13.4 FBPRDY Register (Offset = 22h) [reset = 0h]**

FBPRDY is shown in [Figure 3-161](#) and described in [Table 3-181](#).

Return to the [Summary Table](#).

Flash Bank Pump Ready Register

**Figure 3-161. FBPRDY 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
PUMPRDY	RESERVED						
R-0h	R-0h						
7	6	5	4	3	2	1	0
RESERVED						BANK1RDY	BANK0RDY
R-0h						R-0h	R-0h

**Table 3-181. FBPRDY Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	PUMPRDY	R	0h	Pump Ready. This is a read-only bit which allows software to determine if the pump is ready for flash access before attempting the actual access. If an access is made to a bank when the pump is not ready, wait states are asserted until it becomes ready. 0 Pump is not ready. 1 Pump is ready, in active power state. Reset type: SYSRSn
14-2	RESERVED	R	0h	Reserved
1	BANK1RDY	R	0h	Bank 1 Ready. This is a read-only register which allows software to determine if the Bank 1 is ready for Flash access before the access is attempted. Note: The user should wait for both the pump and the bank to be ready before attempting an access. 0 Bank 1 is not ready. 1 Bank 1 is in active power mode and is ready for access. Reset type: SYSRSn
0	BANK0RDY	R	0h	Bank 0 Ready. This is a read-only register which allows software to determine if the Bank 0 is ready for Flash access before the access is attempted. Note: The user should wait for both the pump and the bank to be ready before attempting an access. 0 Bank 0 is not ready. 1 Bank 0 is in active power mode and is ready for access. Reset type: SYSRSn

**3.14.13.5 FPAC1 Register (Offset = 24h) [reset = 00A0000h]**

 FPAC1 is shown in [Figure 3-162](#) and described in [Table 3-182](#).

 Return to the [Summary Table](#).

Flash Pump Access Control Register 1

**Figure 3-162. FPAC1 Register**

31	30	29	28	27	26	25	24
RESERVED				PSLEEP			
R-0h				R/W-A0h			
23	22	21	20	19	18	17	16
PSLEEP							
R/W-A0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							PMPPWR
R-0h							R/W-0h

**Table 3-182. FPAC1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-16	PSLEEP	R/W	A0h	Pump sleep. These bits contain the starting count value for the charge pump sleep down counter. While the charge pump is in sleep mode, the power mode management logic holds the charge pump sleep counter at this value. When the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP prescaled SYSCLK clock cycles before putting the charge pump into active power mode.  Note: The pump sleep down counter uses the same prescaled clock as Bank sleep down counter which is divided by 2 of input SYSCLK.  Note: BootROM configures the PSLEEP value as 0x3E8 for 100 MHz operation. Users can modify the PSLEEP value based on their application requirements if needed.  Reset type: SYSRSn
15-1	RESERVED	R	0h	Reserved
0	PMPPWR	R/W	0h	Flash Charge Pump Fallback Power Mode. This bit selects what power mode the charge pump enters after the pump active grace period (PAGP) counter has timed out.  0 Sleep (all pump circuits disabled) 1 Active (all pump circuits active)  Note for devices with multiple flash banks: As the pump is shared between flash banks, if an access is made either bank, the value of this bit changes to 1 (active).  Reset type: SYSRSn

### 3.14.13.6 FPAC2 Register (Offset = 26h) [reset = 0h]

FPAC2 is shown in [Figure 3-163](#) and described in [Table 3-183](#).

Return to the [Summary Table](#).

Flash Pump Access Control Register 2

**Figure 3-163. FPAC2 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																PAGP															
R-0h																R/W-0h															

**Table 3-183. FPAC2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	PAGP	R/W	0h	<p>Pump Active Grace Period. This register contains the starting count value for the PAGP mode down counter. Any access to flash memory causes the counter to reload with the PAGP value. After the last access to flash memory, the down counter delays from 0 to 65535 prescaled SYSCLK clock cycles before entering one of the charge pump fallback power modes as determined by PUMPPWR in the FPAC1 register.</p> <p>Note: The PAGP down counter is clocked by the same prescaled clock as the BAGP down counter which is divided by 16 of input SYSCLK.</p> <p>Reset type: SYSRSn</p>

**3.14.13.7 FMSTAT Register (Offset = 2Ah) [reset = 0h]**

 FMSTAT is shown in [Figure 3-164](#) and described in [Table 3-184](#).

 Return to the [Summary Table](#).

Flash Module Status Register

**Figure 3-164. FMSTAT Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED						RESERVED	RESERVED
R-0h						R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	PGV	RESERVED	EV	RESERVED	BUSY
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
ERS	PGM	INVDAT	CSTAT	VOLTSTAT	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-184. FMSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Reserved
17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15	RESERVED	R	0h	Reserved
14	RESERVED	R	0h	Reserved
13	RESERVED	R	0h	Reserved
12	PGV	R	0h	Program verify When set, indicates that a word is not successfully programmed after the maximum allowed number of program pulses are given for program operation. Reset type: SYSRSn
11	RESERVED	R	0h	Reserved
10	EV	R	0h	Erase verify When set, indicates that a sector is not successfully erased after the maximum allowed number of erase pulses are given for erase operation. Reset type: SYSRSn
9	RESERVED	R	0h	Reserved
8	BUSY	R	0h	When set, this bit indicates that a program, erase, or suspend operation is being processed. Reset type: SYSRSn
7	ERS	R	0h	Erase Active. When set, this bit indicates that the flash module is actively performing an erase operation. This bit is set when erasing starts and is cleared when erasing is complete. It is also cleared when the erase is suspended and set when the erase resumes. Reset type: SYSRSn
6	PGM	R	0h	Program Active. When set, this bit indicates that the flash module is currently performing a program operation. This bit is set when programming starts and is cleared when programming is complete. It is also cleared when programming is suspended and set when programming is resumed. Reset type: SYSRSn

**Table 3-184. FMSTAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	INVSTAT	R	0h	Invalid Data. When set, this bit indicates that the user attempted to program a "1" where a "0" was already present. Reset type: SYSRSn
4	CSTAT	R	0h	Command Status. Once the FSM starts any failure will set this bit. When set, this bit informs the host that the program, erase, or validate sector command failed and the command was stopped. This bit is cleared by the Clear Status command. For some errors, this will be the only indication of an FSM error because the cause does not fall within the other error bit types. Reset type: SYSRSn
3	VOLTSTAT	R	0h	Core Voltage Status. When set, this bit indicates that the core voltage generator of the pump power supply dipped below the lower limit allowable during a program or erase operation. Reset type: SYSRSn
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

**3.14.13.8 FRD\_INTF\_CTRL Register (Offset = 180h) [reset = 0h]**

FRD\_INTF\_CTRL is shown in [Figure 3-165](#) and described in [Table 3-185](#).

Return to the [Summary Table](#).

Flash Read Interface Control Register

**Figure 3-165. FRD\_INTF\_CTRL 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						DATA_CACHE_EN	PREFETCH_EN
R-0h						R/W-0h	R/W-0h

**Table 3-185. FRD\_INTF\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-2	RESERVED	R	0h	Reserved
1	DATA_CACHE_EN	R/W	0h	Data cache enable. 0 A value of 0 disables the data cache. 1 A value of 1 enables the data cache. Reset type: SYSRSn
0	PREFETCH_EN	R/W	0h	Prefetch enable. 0 A value of 0 disables prefetch mechanism. 1 A value of 1 enables pre-fetch mechanism. Reset type: SYSRSn



### 3.14.14 FLASH\_ECC\_REGS Registers

Table 3-186 lists the FLASH\_ECC\_REGS registers. All register offset addresses not listed in Table 3-186 should be considered as reserved locations and the register contents should not be modified.

**Table 3-186. FLASH\_ECC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	ECC_ENABLE	ECC Enable	EALLOW	<a href="#">Go</a>
2h	SINGLE_ERR_ADDR_LOW	Single Error Address Low	EALLOW	<a href="#">Go</a>
4h	SINGLE_ERR_ADDR_HIGH	Single Error Address High	EALLOW	<a href="#">Go</a>
6h	UNC_ERR_ADDR_LOW	Uncorrectable Error Address Low	EALLOW	<a href="#">Go</a>
8h	UNC_ERR_ADDR_HIGH	Uncorrectable Error Address High	EALLOW	<a href="#">Go</a>
Ah	ERR_STATUS	Error Status	EALLOW	<a href="#">Go</a>
Ch	ERR_POS	Error Position	EALLOW	<a href="#">Go</a>
Eh	ERR_STATUS_CLR	Error Status Clear	EALLOW	<a href="#">Go</a>
10h	ERR_CNT	Error Control	EALLOW	<a href="#">Go</a>
12h	ERR_THRESHOLD	Error Threshold	EALLOW	<a href="#">Go</a>
14h	ERR_INTFLG	Error Interrupt Flag	EALLOW	<a href="#">Go</a>
16h	ERR_INTCLR	Error Interrupt Flag Clear	EALLOW	<a href="#">Go</a>
18h	FDATAH_TEST	Data High Test	EALLOW	<a href="#">Go</a>
1Ah	FDATAL_TEST	Data Low Test	EALLOW	<a href="#">Go</a>
1Ch	FADDR_TEST	ECC Test Address	EALLOW	<a href="#">Go</a>
1Eh	FECC_TEST	ECC Test Address	EALLOW	<a href="#">Go</a>
20h	FECC_CTRL	ECC Control	EALLOW	<a href="#">Go</a>
22h	FOUTH_TEST	Test Data Out High	EALLOW	<a href="#">Go</a>
24h	FOUTL_TEST	Test Data Out Low	EALLOW	<a href="#">Go</a>
26h	FECC_STATUS	ECC Status	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-187 shows the codes that are used for access types in this section.

**Table 3-187. FLASH\_ECC\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.

**Table 3-187. FLASH\_ECC\_REGS Access Type Codes (continued)**

Access Type	Code	Description
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.14.1 ECC\_ENABLE Register (Offset = 0h) [reset = Ah]**

ECC\_ENABLE is shown in [Figure 3-166](#) and described in [Table 3-188](#).

Return to the [Summary Table](#).

ECC Enable

**Figure 3-166. ECC\_ENABLE 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												ENABLE			
R-0h												R/W-Ah			

**Table 3-188. ECC\_ENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3-0	ENABLE	R/W	Ah	ECC enable. A value of 0xA would enable ECC. Any other value would disable ECC. Reset type: SYSRSn

**3.14.14.2 SINGLE\_ERR\_ADDR\_LOW Register (Offset = 2h) [reset = 0h]**

SINGLE\_ERR\_ADDR\_LOW is shown in [Figure 3-167](#) and described in [Table 3-189](#).

Return to the [Summary Table](#).

Single Error Address Low

**Figure 3-167. SINGLE\_ERR\_ADDR\_LOW 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
ERR_ADDR_L																															
R/W-0h																															

**Table 3-189. SINGLE\_ERR\_ADDR\_LOW Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ERR_ADDR_L	R/W	0h	64-bit aligned address at which a single bit error occurred in the lower 64-bits of a 128-bit aligned memory. Reset type: SYSRSn

**3.14.14.3 SINGLE\_ERR\_ADDR\_HIGH Register (Offset = 4h) [reset = 0h]**

SINGLE\_ERR\_ADDR\_HIGH is shown in [Figure 3-168](#) and described in [Table 3-190](#).

Return to the [Summary Table](#).

Single Error Address High

**Figure 3-168. SINGLE\_ERR\_ADDR\_HIGH 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
ERR_ADDR_H																															
R/W-0h																															

**Table 3-190. SINGLE\_ERR\_ADDR\_HIGH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ERR_ADDR_H	R/W	0h	64-bit aligned address at which a single bit error occurred in the upper 64-bits of a 128-bit aligned memory. Reset type: SYSRSn

**3.14.14.4 UNC\_ERR\_ADDR\_LOW Register (Offset = 6h) [reset = 0h]**

UNC\_ERR\_ADDR\_LOW is shown in [Figure 3-169](#) and described in [Table 3-191](#).

Return to the [Summary Table](#).

Uncorrectable Error Address Low

**Figure 3-169. UNC\_ERR\_ADDR\_LOW 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
UNC_ERR_ADDR_L																															
R/W-0h																															

**Table 3-191. UNC\_ERR\_ADDR\_LOW Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	UNC_ERR_ADDR_L	R/W	0h	64-bit aligned address at which an uncorrectable error occurred in the lower 64-bits of a 128-bit aligned memory. Reset type: SYSRSn

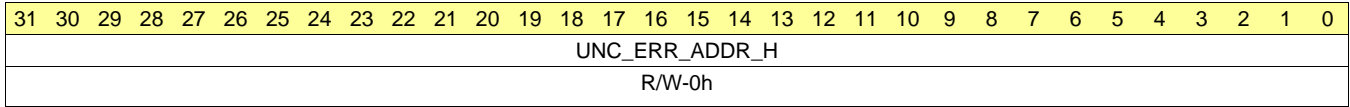
**3.14.14.5 UNC\_ERR\_ADDR\_HIGH Register (Offset = 8h) [reset = 0h]**

UNC\_ERR\_ADDR\_HIGH is shown in [Figure 3-170](#) and described in [Table 3-192](#).

Return to the [Summary Table](#).

Uncorrectable Error Address High

**Figure 3-170. UNC\_ERR\_ADDR\_HIGH Register**



**Table 3-192. UNC\_ERR\_ADDR\_HIGH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	UNC_ERR_ADDR_H	R/W	0h	64-bit aligned address at which an uncorrectable error occurred in the upper 64-bits of a 128-bit aligned memory. Reset type: SYSRSn

**3.14.14.6 ERR\_STATUS Register (Offset = Ah) [reset = 0h]**

ERR\_STATUS is shown in [Figure 3-171](#) and described in [Table 3-193](#).

Return to the [Summary Table](#).

Error Status

**Figure 3-171. ERR\_STATUS Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED					UNC_ERR_H	FAIL_1_H	FAIL_0_H
R-0h					R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					UNC_ERR_L	FAIL_1_L	FAIL_0_L
R-0h					R-0h	R-0h	R-0h

**Table 3-193. ERR\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18	UNC_ERR_H	R	0h	Uncorrectable error. A value of 1 indicates that an un-correctable error occurred in upper 64bits of a 128-bit aligned address. Cleared by writing a 1 to UNC_ERR_H_CLR bit of ERR_STATUS_CLR register. Reset type: SYSRSn
17	FAIL_1_H	R	0h	Fail on 1. 0 Fail on 1 single bit error did not occur in upper 64bits of a 128-bit aligned address. 1 A value of 1 would indicate that a single bit error occurred in upper 64bits of a 128-bit aligned address and the corrected value was 1. Cleared by writing a 1 to FAIL_1_H_CLR bit of ERR_STATUS_CLR register. Note: This bit is updated on every flash access which results in a single bit error, So, in case of multiple single bit error, the status would correspond to the last error which occurred. Reset type: SYSRSn
16	FAIL_0_H	R	0h	Fail on 0. 0 Fail on 0 single bit error did not occur in upper 64bits of a 128-bit aligned address. 1 A value of 1 would indicate that a single bit error occurred in upper 64bits of a 128-bit aligned address and the corrected value was 0. Cleared by writing a 1 to FAIL_0_H_CLR bit of ERR_STATUS_CLR register. Note: This bit is updated on every flash access which results in a single bit error, So, in case of multiple single bit error, the status would correspond to the last error which occurred. Reset type: SYSRSn
15-3	RESERVED	R	0h	Reserved
2	UNC_ERR_L	R	0h	Uncorrectable error. A value of 1 indicates that an un-correctable error occurred in lower 64bits of a 128-bit aligned address. Cleared by writing a 1 to UNC_ERR_L_CLR bit of ERR_STATUS_CLR register. Reset type: SYSRSn



**Table 3-193. ERR\_STATUS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	FAIL_1_L	R	0h	<p>Fail on 1.</p> <p>0 Fail on 1 single bit error did not occur in lower 64bits of a 128-bit aligned address.</p> <p>1 A value of 1 would indicate that a single bit error occurred in lower 64bits of a 128-bit aligned address and the corrected value was 1. Cleared by writing a 1 to FAIL_1_L_CLR bit of ERR_STATUS_CLR register.</p> <p>Note: This bit is updated on every flash access which results in a single bit error, So, in case of multiple single bit error, the status would correspond to the last error which occurred.</p> <p>Reset type: SYSRSn</p>
0	FAIL_0_L	R	0h	<p>Fail on 0.</p> <p>0 Fail on 0 single bit error did not occur in lower 64bits of a 128-bit aligned address.</p> <p>1 Would indicate that a single bit error occurred in lower 64bits of a 128-bit aligned address and the corrected value was 0. Cleared by writing a 1 to FAIL_0_L_CLR bit of ERR_STATUS_CLR register.</p> <p>Note: This bit is updated on every flash access which results in a single bit error, So, in case of multiple single bit error, the status would correspond to the last error which occurred.</p> <p>Reset type: SYSRSn</p>

**3.14.14.7 ERR\_POS Register (Offset = Ch) [reset = 0h]**

ERR\_POS is shown in [Figure 3-172](#) and described in [Table 3-194](#).

Return to the [Summary Table](#).

Error Position

**Figure 3-172. ERR\_POS Register**

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

**Table 3-194. ERR\_POS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	ERR_TYPE_H	R	0h	Error type 0 Indicates that a single bit error occurred in upper 64 data bits of a 128-bit aligned address. 1 Indicates that a single bit error occurred in ECC check bits of upper 64bits of a 128-bit aligned address. Reset type: SYSRSn
23-22	RESERVED	R	0h	Reserved
21-16	ERR_POS_H	R	0h	Error position. Bit position of the single bit error in upper 64bits of a 128-bit aligned address. The position is interpreted depending on whether the ERR_TYPE bit indicates a check bit or a data bit. If ERR_TYPE indicates a check bit error, the error position could range from 0 to 7, else it could range from 0 to 63. Reset type: SYSRSn
15-9	RESERVED	R	0h	Reserved
8	ERR_TYPE_L	R	0h	Error type 0 Indicates that a single bit error occurred in lower 64 data bits of a 128-bit aligned address. 1 Indicates that a single bit error occurred in ECC check bits of lower 64bits of a 128-bit aligned address. Reset type: SYSRSn
7-6	RESERVED	R	0h	Reserved
5-0	ERR_POS_L	R	0h	Error position. Bit position of the single bit error in lower 64bits of a 128-bit aligned address. The position is interpreted depending on whether the ERR_TYPE bit indicates a check bit or a data bit. If ERR_TYPE indicates a check bit error, the error position could range from 0 to 7, else it could range from 0 to 63. Reset type: SYSRSn

### 3.14.14.8 ERR\_STATUS\_CLR Register (Offset = Eh) [reset = 0h]

ERR\_STATUS\_CLR is shown in [Figure 3-173](#) and described in [Table 3-195](#).

Return to the [Summary Table](#).

Error Status Clear

**Figure 3-173. ERR\_STATUS\_CLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED					UNC_ERR_H_CLR	FAIL_1_H_CLR	FAIL_0_H_CLR
R-0h					R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					UNC_ERR_L_CLR	FAIL_1_L_CLR	FAIL_0_L_CLR
R-0h					R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-195. ERR\_STATUS\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18	UNC_ERR_H_CLR	R-0/W1S	0h	Uncorrectable error clear. Writing a 1 to this bit will clear the UNC_ERR_H bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0. Reset type: SYSRSn
17	FAIL_1_H_CLR	R-0/W1S	0h	Fail on 1 clear. Writing a 1 to this bit will clear the FAIL_1_H bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0. Reset type: SYSRSn
16	FAIL_0_H_CLR	R-0/W1S	0h	Fail on 0 clear. Writing a 1 to this bit will clear the FAIL_0_H bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0. Reset type: SYSRSn
15-3	RESERVED	R	0h	Reserved
2	UNC_ERR_L_CLR	R-0/W1S	0h	Uncorrectable error clear. Writing a 1 to this bit will clear the UNC_ERR_L bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0. Reset type: SYSRSn
1	FAIL_1_L_CLR	R-0/W1S	0h	Fail on 1 clear. Writing a 1 to this bit will clear the FAIL_1_L bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0. Reset type: SYSRSn
0	FAIL_0_L_CLR	R-0/W1S	0h	Fail on 0 clear. Writing a 1 to this bit will clear the FAIL_0_L bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0. Reset type: SYSRSn

**3.14.14.9 ERR\_CNT Register (Offset = 10h) [reset = 0h]**

ERR\_CNT is shown in [Figure 3-174](#) and described in [Table 3-196](#).

Return to the [Summary Table](#).

Error Control

**Figure 3-174. ERR\_CNT 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																ERR_CNT															
R-0h																R/W-0h															

**Table 3-196. ERR\_CNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	ERR_CNT	R/W	0h	Single bit error count. This counter increments with every single bit ECC error occurrence. Upon reaching the threshold value counter stops counting on single bit errors. ERR_CNT can be cleared (irrespective of whether threshold is met or not) using "Single Err Int Clear" bit. This is applicable for ECC logic test mode and normal operational mode. Reset type: SYSRSn

**3.14.14.10 ERR\_THRESHOLD Register (Offset = 12h) [reset = 0h]**

ERR\_THRESHOLD is shown in [Figure 3-175](#) and described in [Table 3-197](#).

Return to the [Summary Table](#).

Error Threshold

**Figure 3-175. ERR\_THRESHOLD 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																ERR_THRESHOLD															
R-0h																R/W-0h															

**Table 3-197. ERR\_THRESHOLD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	ERR_THRESHOLD	R/W	0h	Single bit error threshold. Sets the threshold for single bit errors. When the ERR_CNT value equals the THRESHOLD value and a single bit error occurs, SINGLE_ERR_INT flag is set, and an interrupt is fired.  Reset type: SYSRSn

**3.14.14.11 ERR\_INTFLG Register (Offset = 14h) [reset = 0h]**

ERR\_INTFLG is shown in [Figure 3-176](#) and described in [Table 3-198](#).

Return to the [Summary Table](#).

Error Interrupt Flag

**Figure 3-176. ERR\_INTFLG 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						UNC_ERR_INT FLG	SINGLE_ERR_ INTFLG
R-0h						R-0h	R-0h

**Table 3-198. ERR\_INTFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-2	RESERVED	R	0h	Reserved
1	UNC_ERR_INTFLG	R	0h	Uncorrectable bit error interrupt flag. When a Un-correctable error occurs, this bit is set and the UNC_ERR_INT interrupt is fired. When UNC_ERR_INTCLR bit of ERR_INTCLR register is written a value of 1 this bit is cleared. Reset type: SYSRSn
0	SINGLE_ERR_INTFLG	R	0h	Single bit error interrupt flag. When the ERR_CNT value equals the ERR_THRESHOLD value and a single bit error occurs then SINGLE_ERR_INT flag is set and SINGLE_ERR_INT interrupt is fired. When SINGLE_ERR_INTCLR bit of ERR_INTCLR register is written a value of 1 this bit is cleared. Reset type: SYSRSn

**3.14.14.12 ERR\_INTCLR Register (Offset = 16h) [reset = 0h]**

ERR\_INTCLR is shown in [Figure 3-177](#) and described in [Table 3-199](#).

Return to the [Summary Table](#).

Error Interrupt Flag Clear

**Figure 3-177. ERR\_INTCLR 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						UNC_ERR_INT CLR	SINGLE_ERR_ INTCLR
R-0h						R-0/W1S-0h	R-0/W1S-0h

**Table 3-199. ERR\_INTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-2	RESERVED	R	0h	Reserved
1	UNC_ERR_INTCLR	R-0/W1S	0h	Uncorrectable bit error interrupt flag clear. Writing a 1 to this bit will clear UNC_ERR_INT_FLG. Writes of 0 have no effect. Reset type: SYSRSn
0	SINGLE_ERR_INTCLR	R-0/W1S	0h	Single bit error interrupt flag clear. Writing a 1 to this bit will clear SINGLE_ERR_INT_FLG. Writes of 0 have no effect. Reset type: SYSRSn

**3.14.14.13 FDATAH\_TEST Register (Offset = 18h) [reset = 0h]**

FDATAH\_TEST is shown in [Figure 3-178](#) and described in [Table 3-200](#).

Return to the [Summary Table](#).

Data High Test

**Figure 3-178. FDATAH\_TEST 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
FDATAH																															
R/W-0h																															

**Table 3-200. FDATAH\_TEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	FDATAH	R/W	0h	High double word of selected 64-bit data. User-configurable bits 63:32 of the selected data block in ECC test mode. Reset type: SYSRSn



**3.14.14.14 FDATA<sub>L</sub>\_TEST Register (Offset = 1Ah) [reset = 0h]**

FDATA<sub>L</sub>\_TEST is shown in [Figure 3-179](#) and described in [Table 3-201](#).

Return to the [Summary Table](#).

Data Low Test

**Figure 3-179. FDATA<sub>L</sub>\_TEST 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
FDATA <sub>L</sub>																															
R/W-0h																															

**Table 3-201. FDATA<sub>L</sub>\_TEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	FDATA <sub>L</sub>	R/W	0h	Low double word of selected 64-bit data. User-configurable bits 31:0 of the selected data block in ECC test mode. Reset type: SYSRSn

**3.14.14.15 FADDR\_TEST Register (Offset = 1Ch) [reset = 0h]**

FADDR\_TEST is shown in [Figure 3-180](#) and described in [Table 3-202](#).

Return to the [Summary Table](#).

ECC Test Address

**Figure 3-180. FADDR\_TEST Register**

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

**Table 3-202. FADDR\_TEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0h	Reserved
21-16	ADDRH	R/W	0h	Address for selected 64-bit data. User-configurable address bits of the selected data in ECC test mode. Left-shift the address by one bit (to provide byte address) and ignore the three least significant bits of the address and write the bits 21:16 in remaining address bits in this field. Reset type: SYSRSn
15-3	ADDRL	R/W	0h	Address for selected 64-bit data. User-configurable address bits of the selected data in ECC test mode. Left-shift the address by one bit (to provide byte address) and ignore the three least significant bits of the address and write the bits 15:3 in remaining address bits in this field. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

**3.14.14.16 FECC\_TEST Register (Offset = 1Eh) [reset = 0h]**

FECC\_TEST is shown in [Figure 3-181](#) and described in [Table 3-203](#).

Return to the [Summary Table](#).

ECC Test Address

**Figure 3-181. FECC\_TEST 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																RESERVED								ECC							
R-0h																R-0h								R/W-0h							

**Table 3-203. FECC\_TEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7-0	ECC	R/W	0h	8-bit ECC for selected 64-bit data. User-configurable ECC bits of the selected 64-bit data block in ECC test mode. Reset type: SYSRSn

**3.14.14.17 FECC\_CTRL Register (Offset = 20h) [reset = 0h]**

 FECC\_CTRL is shown in [Figure 3-182](#) and described in [Table 3-204](#).

 Return to the [Summary Table](#).

ECC Control

**Figure 3-182. FECC\_CTRL 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					DO_ECC_CALC	ECC_SELECT	ECC_TEST_EN
R-0h					R-0/W1S-0h	R/W-0h	R/W-0h

**Table 3-204. FECC\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-3	RESERVED	R	0h	Reserved
2	DO_ECC_CALC	R-0/W1S	0h	Enable ECC calculation. ECC logic will calculate ECC in one cycle for the data and address written in ECC test registers when ECC test logic is enabled by setting ECC_TEST_EN. Reset type: SYSRSn
1	ECC_SELECT	R/W	0h	ECC block select. 0 Selects the ECC block on bits [63:0] of bank data. 1 Selects the ECC block on bits [127:64] of bank data. Reset type: SYSRSn
0	ECC_TEST_EN	R/W	0h	ECC test mode enable. 0 ECC test mode disabled 1 ECC test mode enabled Reset type: SYSRSn

**3.14.14.18 FOUTH\_TEST Register (Offset = 22h) [reset = 0h]**

FOUTH\_TEST is shown in [Figure 3-183](#) and described in [Table 3-205](#).

Return to the [Summary Table](#).

Test Data Out High

**Figure 3-183. FOUTH\_TEST 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
DATAOUTH																															
R-0h																															

**Table 3-205. FOUTH\_TEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DATAOUTH	R	0h	High double word test data out. Holds bits 63:32 of the data out of the selected ECC block. Reset type: SYSRSn

**3.14.14.19 FOUTL\_TEST Register (Offset = 24h) [reset = 0h]**

FOUTL\_TEST is shown in [Figure 3-184](#) and described in [Table 3-206](#).

Return to the [Summary Table](#).

Test Data Out Low

**Figure 3-184. FOUTL\_TEST 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
DATAOUTL																															
R-0h																															

**Table 3-206. FOUTL\_TEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DATAOUTL	R	0h	Low double word test data out. Holds bits 31:0 of the data out of the selected ECC block. Reset type: SYSRSn

**3.14.14.20 FECC\_STATUS Register (Offset = 26h) [reset = 0h]**

 FECC\_STATUS is shown in [Figure 3-185](#) and described in [Table 3-207](#).

 Return to the [Summary Table](#).

ECC Status

**Figure 3-185. FECC\_STATUS 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							ERR_TYPE
R-0h							R-0h
7	6	5	4	3	2	1	0
DATA_ERR_POS						UNC_ERR	SINGLE_ERR
R-0h						R-0h	R-0h

**Table 3-207. FECC\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-9	RESERVED	R	0h	Reserved
8	ERR_TYPE	R	0h	Test mode ECC single bit error indicator. When 1, indicates that the single bit error is in check bits. When 0, indicates that the single bit error is in data bits (If SINGLE_ERR field is also set). Reset type: SYSRSn
7-2	DATA_ERR_POS	R	0h	Test mode single bit error position. Holds the bit position where the single bit error occurred. The position is interpreted depending on whether the CHK_ERR bit indicates a check bit or a data bit. If CHK_ERR indicates a check bit error, the error position could range from 0 to 7, or it could range from 0 to 63. Reset type: SYSRSn
1	UNC_ERR	R	0h	Test mode ECC double bit error. When 1 indicates that the ECC test resulted in an uncorrectable bit error. Reset type: SYSRSn
0	SINGLE_ERR	R	0h	Test mode ECC single bit error. When 1 indicates that the ECC test resulted in a single bit error. Reset type: SYSRSn

### 3.14.15 MEM\_CFG\_REGS Registers

Table 3-208 lists the MEM\_CFG\_REGS registers. All register offset addresses not listed in Table 3-208 should be considered as reserved locations and the register contents should not be modified.

**Table 3-208. MEM\_CFG\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	DxLOCK	Dedicated RAM Config Lock Register	EALLOW	<a href="#">Go</a>
2h	DxCOMMIT	Dedicated RAM Config Lock Commit Register	EALLOW	<a href="#">Go</a>
8h	DxACCPROT0	Dedicated RAM Config Register	EALLOW	<a href="#">Go</a>
10h	DxTEST	Dedicated RAM TEST Register	EALLOW	<a href="#">Go</a>
12h	DxINIT	Dedicated RAM Init Register	EALLOW	<a href="#">Go</a>
14h	DxINITDONE	Dedicated RAM InitDone Status Register		<a href="#">Go</a>
20h	LSxLOCK	Local Shared RAM Config Lock Register	EALLOW	<a href="#">Go</a>
22h	LSxCOMMIT	Local Shared RAM Config Lock Commit Register	EALLOW	<a href="#">Go</a>
24h	LSxMSEL	Local Shared RAM Master Sel Register	EALLOW	<a href="#">Go</a>
26h	LSxCLAPGM	Local Shared RAM Prog/Exe control Register	EALLOW	<a href="#">Go</a>
28h	LSxACCPROT0	Local Shared RAM Config Register 0	EALLOW	<a href="#">Go</a>
2Ah	LSxACCPROT1	Local Shared RAM Config Register 1	EALLOW	<a href="#">Go</a>
30h	LSxTEST	Local Shared RAM TEST Register	EALLOW	<a href="#">Go</a>
32h	LSxINIT	Local Shared RAM Init Register	EALLOW	<a href="#">Go</a>
34h	LSxINITDONE	Local Shared RAM InitDone Status Register		<a href="#">Go</a>
40h	GSxLOCK	Global Shared RAM Config Lock Register	EALLOW	<a href="#">Go</a>
42h	GSxCOMMIT	Global Shared RAM Config Lock Commit Register	EALLOW	<a href="#">Go</a>
48h	GSxACCPROT0	Global Shared RAM Config Register 0	EALLOW	<a href="#">Go</a>
4Ah	GSxACCPROT1	Global Shared RAM Config Register 1	EALLOW	<a href="#">Go</a>
4Ch	GSxACCPROT2	Global Shared RAM Config Register 2	EALLOW	<a href="#">Go</a>
4Eh	GSxACCPROT3	Global Shared RAM Config Register 3	EALLOW	<a href="#">Go</a>
50h	GSxTEST	Global Shared RAM TEST Register	EALLOW	<a href="#">Go</a>
52h	GSxINIT	Global Shared RAM Init Register	EALLOW	<a href="#">Go</a>
54h	GSxINITDONE	Global Shared RAM InitDone Status Register		<a href="#">Go</a>
60h	MSGxLOCK	Message RAM Config Lock Register		<a href="#">Go</a>
62h	MSGxCOMMIT	Message RAM Config Lock Commit Register		<a href="#">Go</a>
70h	MSGxTEST	Message RAM TEST Register	EALLOW	<a href="#">Go</a>
72h	MSGxINIT	Message RAM Init Register	EALLOW	<a href="#">Go</a>
74h	MSGxINITDONE	Message RAM InitDone Status Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-209 shows the codes that are used for access types in this section.

**Table 3-209. MEM\_CFG\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set



**Table 3-209. MEM\_CFG\_REGS Access Type Codes (continued)**

Access Type	Code	Description
WSonce	W Sonce	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.15.1 DxLOCK Register (Offset = 0h) [reset = 0h]**

DxLOCK is shown in [Figure 3-186](#) and described in [Table 3-210](#).

Return to the [Summary Table](#).

Dedicated RAM Config Lock Register

**Figure 3-186. DxLOCK 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				RESERVED	RESERVED	LOCK_M1	LOCK_M0
R-0h						R/W-0h	R/W-0h

**Table 3-210. DxLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	LOCK_M1	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for M1 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked. Reset type: SYSRSn
0	LOCK_M0	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for M0 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked. Reset type: SYSRSn

**3.14.15.2 DxCOMMIT Register (Offset = 2h) [reset = 0h]**

DxCOMMIT is shown in [Figure 3-187](#) and described in [Table 3-211](#).

Return to the [Summary Table](#).

Dedicated RAM Config Lock Commit Register

**Figure 3-187. DxCOMMIT 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				RESERVED	RESERVED	COMMIT_M1	COMMIT_M0
R-0h						R/WOnce-0h	R/WOnce-0h

**Table 3-211. DxCOMMIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R/WOnce	0h	Reserved
2	RESERVED	R/WOnce	0h	Reserved
1	COMMIT_M1	R/WOnce	0h	Permanently Locks the write to access protection, master select, initialization control and test register fields for M1 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in DxLOCK register. 1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked. Reset type: SYSRSn
0	COMMIT_M0	R/WOnce	0h	Permanently Locks the write to access protection, master select, initialization control and test register fields for M0 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in DxLOCK register. 1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked. Reset type: SYSRSn

### 3.14.15.3 DxACCPROT0 Register (Offset = 8h) [reset = 0h]

DxACCPROT0 is shown in [Figure 3-188](#) and described in [Table 3-212](#).

Return to the [Summary Table](#).

Dedicated RAM Config Register

**Figure 3-188. DxACCPROT0 Register**

31	30	29	28	27	26	25	24
RESERVED						RESERVED	RESERVED
R-0h							
23	22	21	20	19	18	17	16
RESERVED						RESERVED	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 3-212. DxACCPROT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23-18	RESERVED	R	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-0	RESERVED	R	0h	Reserved

**3.14.15.4 DxTEST Register (Offset = 10h) [reset = 0h]**

DxTEST is shown in [Figure 3-189](#) and described in [Table 3-213](#).

Return to the [Summary Table](#).

Dedicated RAM TEST Register

**Figure 3-189. DxTEST 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		RESERVED		TEST_M1		TEST_M0	
				R/W-0h		R/W-0h	

**Table 3-213. DxTEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7-6	RESERVED	R/W	0h	Reserved
5-4	RESERVED	R/W	0h	Reserved
3-2	TEST_M1	R/W	0h	Selects the defferent modes for M1 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to ECC bits. 10: Writes are allowed to ECC bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
1-0	TEST_M0	R/W	0h	Selects the defferent modes for M0 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to ECC bits. 10: Writes are allowed to ECC bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn

**3.14.15.5 DxINIT Register (Offset = 12h) [reset = 0h]**

DxINIT is shown in [Figure 3-190](#) and described in [Table 3-214](#).

Return to the [Summary Table](#).

Dedicated RAM Init Register

**Figure 3-190. DxINIT 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				RESERVED	RESERVED	INIT_M1	INIT_M0
R-0h						R-0/W1S-0h	R-0/W1S-0h

**Table 3-214. DxINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	RESERVED	R-0/W1S	0h	Reserved
1	INIT_M1	R-0/W1S	0h	RAM Initialization control for M1 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
0	INIT_M0	R-0/W1S	0h	RAM Initialization control for M0 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn

**3.14.15.6 DxINITDONE Register (Offset = 14h) [reset = 0h]**

DxINITDONE is shown in [Figure 3-191](#) and described in [Table 3-215](#).

Return to the [Summary Table](#).

Dedicated RAM InitDone Status Register

**Figure 3-191. DxINITDONE 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				RESERVED	RESERVED	INITDONE_M1	INITDONE_M0
R-0h						R-0h	R-0h

**Table 3-215. DxINITDONE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	INITDONE_M1	R	0h	RAM Initialization status for M1 RAM: 0: RAM Initialization is not done. 1: RAM Initialization has completed. Reset type: SYSRSn
0	INITDONE_M0	R	0h	RAM Initialization status for M0 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn

**3.14.15.7 LSxLOCK Register (Offset = 20h) [reset = 0h]**

LSxLOCK is shown in [Figure 3-192](#) and described in [Table 3-216](#).

Return to the [Summary Table](#).

Local Shared RAM Config Lock Register

**Figure 3-192. LSxLOCK 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
LOCK_LS7	LOCK_LS6	LOCK_LS5	LOCK_LS4	LOCK_LS3	LOCK_LS2	LOCK_LS1	LOCK_LS0
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 3-216. LSxLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	LOCK_LS7	R/W	0h	Locks the write to access protection, master select, program or data memory select, initialization control and register fields test for LS7 RAM: 0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked. Reset type: SYSRSn
6	LOCK_LS6	R/W	0h	Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS6 RAM: 0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked. Reset type: SYSRSn
5	LOCK_LS5	R/W	0h	Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS5 RAM: 0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked. Reset type: SYSRSn



**Table 3-216. LSxLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	LOCK_LS4	R/W	0h	<p>Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS4 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.</p> <p>Reset type: SYSRSn</p>
3	LOCK_LS3	R/W	0h	<p>Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS3 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.</p> <p>Reset type: SYSRSn</p>
2	LOCK_LS2	R/W	0h	<p>Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS2 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.</p> <p>Reset type: SYSRSn</p>
1	LOCK_LS1	R/W	0h	<p>Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS1 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.</p> <p>Reset type: SYSRSn</p>
0	LOCK_LS0	R/W	0h	<p>Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS0 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.</p> <p>Reset type: SYSRSn</p>

**3.14.15.8 LSxCOMMIT Register (Offset = 22h) [reset = 0h]**

LSxCOMMIT is shown in [Figure 3-193](#) and described in [Table 3-217](#).

Return to the [Summary Table](#).

Local Shared RAM Config Lock Commit Register

**Figure 3-193. LSxCOMMIT 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
COMMIT_LS7	COMMIT_LS6	COMMIT_LS5	COMMIT_LS4	COMMIT_LS3	COMMIT_LS2	COMMIT_LS1	COMMIT_LS0
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 3-217. LSxCOMMIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	COMMIT_LS7	R/WOnce	0h	Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS7 RAM: 0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked. Reset type: SYSRSn
6	COMMIT_LS6	R/WOnce	0h	Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS6 RAM: 0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked. Reset type: SYSRSn
5	COMMIT_LS5	R/WOnce	0h	Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS5 RAM: 0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked. Reset type: SYSRSn

**Table 3-217. LSxCOMMIT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	COMMIT_LS4	R/WOnce	0h	<p>Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS4 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.</p> <p>Reset type: SYSRSn</p>
3	COMMIT_LS3	R/WOnce	0h	<p>Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS3 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.</p> <p>Reset type: SYSRSn</p>
2	COMMIT_LS2	R/WOnce	0h	<p>Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS2 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.</p> <p>Reset type: SYSRSn</p>
1	COMMIT_LS1	R/WOnce	0h	<p>Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS1 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.</p> <p>Reset type: SYSRSn</p>
0	COMMIT_LS0	R/WOnce	0h	<p>Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS0 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.</p> <p>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.</p> <p>Reset type: SYSRSn</p>

### 3.14.15.9 LSxMSEL Register (Offset = 24h) [reset = 0h]

LSxMSEL is shown in [Figure 3-194](#) and described in [Table 3-218](#).

Return to the [Summary Table](#).

Local Shared RAM Master Sel Register

**Figure 3-194. LSxMSEL 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
MSEL_LS7		MSEL_LS6		MSEL_LS5		MSEL_LS4	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
MSEL_LS3		MSEL_LS2		MSEL_LS1		MSEL_LS0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 3-218. LSxMSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	MSEL_LS7	R/W	0h	Master Select for LS7 RAM: 00: Memory is dedicated to CPU. 01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'. 10: Reserved. 11: Reserved. Reset type: SYSRSn
13-12	MSEL_LS6	R/W	0h	Master Select for LS6 RAM: 00: Memory is dedicated to CPU. 01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'. 10: Reserved. 11: Reserved. Reset type: SYSRSn
11-10	MSEL_LS5	R/W	0h	Master Select for LS5 RAM: 00: Memory is dedicated to CPU. 01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'. 10: Reserved. 11: Reserved. Reset type: SYSRSn
9-8	MSEL_LS4	R/W	0h	Master Select for LS4 RAM: 00: Memory is dedicated to CPU. 01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'. 10: Reserved. 11: Reserved. Reset type: SYSRSn

**Table 3-218. LSxMSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	MSEL_LS3	R/W	0h	Master Select for LS3 RAM: 00: Memory is dedicated to CPU. 01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'. 10: Reserved. 11: Reserved. Reset type: SYSRSn
5-4	MSEL_LS2	R/W	0h	Master Select for LS2 RAM: 00: Memory is dedicated to CPU. 01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'. 10: Reserved. 11: Reserved. Reset type: SYSRSn
3-2	MSEL_LS1	R/W	0h	Master Select for LS1 RAM: 00: Memory is dedicated to CPU. 01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'. 10: Reserved. 11: Reserved. Reset type: SYSRSn
1-0	MSEL_LS0	R/W	0h	Master Select for LS0 RAM: 00: Memory is dedicated to CPU. 01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'. 10: Reserved. 11: Reserved. Reset type: SYSRSn

**3.14.15.10 LSxCLAPGM Register (Offset = 26h) [reset = 0h]**

LSxCLAPGM is shown in [Figure 3-195](#) and described in [Table 3-219](#).

Return to the [Summary Table](#).

Local Shared RAM Prog/Exe control Register

**Figure 3-195. LSxCLAPGM 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
CLAPGM_LS7	CLAPGM_LS6	CLAPGM_LS5	CLAPGM_LS4	CLAPGM_LS3	CLAPGM_LS2	CLAPGM_LS1	CLAPGM_LS0
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 3-219. LSxCLAPGM Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	CLAPGM_LS7	R/W	0h	Selects LS7 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory. Reset type: SYSRSn
6	CLAPGM_LS6	R/W	0h	Selects LS6 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory. Reset type: SYSRSn
5	CLAPGM_LS5	R/W	0h	Selects LS5 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory. Reset type: SYSRSn
4	CLAPGM_LS4	R/W	0h	Selects LS4 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory. Reset type: SYSRSn
3	CLAPGM_LS3	R/W	0h	Selects LS3 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory. Reset type: SYSRSn
2	CLAPGM_LS2	R/W	0h	Selects LS2 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory. Reset type: SYSRSn

**Table 3-219. LSxCLAPGM Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	CLAPGM_LS1	R/W	0h	Selects LS1 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory. Reset type: SYSRSn
0	CLAPGM_LS0	R/W	0h	Selects LS0 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory. Reset type: SYSRSn

**3.14.15.11 LSxACCPROT0 Register (Offset = 28h) [reset = 0h]**

 LSxACCPROT0 is shown in [Figure 3-196](#) and described in [Table 3-220](#).

 Return to the [Summary Table](#).

Local Shared RAM Config Register 0

**Figure 3-196. LSxACCPROT0 Register**

31	30	29	28	27	26	25	24
RESERVED						CPUWRPROT_ LS3	FETCHPROT_ LS3
R-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED						CPUWRPROT_ LS2	FETCHPROT_ LS2
R-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED						CPUWRPROT_ LS1	FETCHPROT_ LS1
R-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						CPUWRPROT_ LS0	FETCHPROT_ LS0
R-0h						R/W-0h	R/W-0h

**Table 3-220. LSxACCPROT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	CPUWRPROT_LS3	R/W	0h	CPU WR Protection For LS3 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
24	FETCHPROT_LS3	R/W	0h	Fetch Protection For LS3 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
23-18	RESERVED	R	0h	Reserved
17	CPUWRPROT_LS2	R/W	0h	CPU WR Protection For LS2 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
16	FETCHPROT_LS2	R/W	0h	Fetch Protection For LS2 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
15-10	RESERVED	R	0h	Reserved
9	CPUWRPROT_LS1	R/W	0h	CPU WR Protection For LS1 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn



**Table 3-220. LSxACCPROT0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	FETCHPROT_LS1	R/W	0h	Fetch Protection For LS1 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	CPUWRPROT_LS0	R/W	0h	CPU WR Protection For LS0 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
0	FETCHPROT_LS0	R/W	0h	Fetch Protection For LS0 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn

**3.14.15.12 LSxACCPROT1 Register (Offset = 2Ah) [reset = 0h]**

LSxACCPROT1 is shown in [Figure 3-197](#) and described in [Table 3-221](#).

Return to the [Summary Table](#).

Local Shared RAM Config Register 1

**Figure 3-197. LSxACCPROT1 Register**

31	30	29	28	27	26	25	24
RESERVED						CPUWRPROT_ LS7	FETCHPROT_ LS7
R-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED						CPUWRPROT_ LS6	FETCHPROT_ LS6
R-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED						CPUWRPROT_ LS5	FETCHPROT_ LS5
R-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						CPUWRPROT_ LS4	FETCHPROT_ LS4
R-0h						R/W-0h	R/W-0h

**Table 3-221. LSxACCPROT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	CPUWRPROT_LS7	R/W	0h	CPU WR Protection For LS7 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
24	FETCHPROT_LS7	R/W	0h	Fetch Protection For LS7 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
23-18	RESERVED	R	0h	Reserved
17	CPUWRPROT_LS6	R/W	0h	CPU WR Protection For LS6 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
16	FETCHPROT_LS6	R/W	0h	Fetch Protection For LS6 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
15-10	RESERVED	R	0h	Reserved
9	CPUWRPROT_LS5	R/W	0h	CPU WR Protection For LS5 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn

**Table 3-221. LSxACCPROT1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	FETCHPROT_LS5	R/W	0h	Fetch Protection For LS5 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	CPUWRPROT_LS4	R/W	0h	CPU WR Protection For LS4 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
0	FETCHPROT_LS4	R/W	0h	Fetch Protection For LS4 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn

**3.14.15.13 LSxTEST Register (Offset = 30h) [reset = 0h]**

LSxTEST is shown in [Figure 3-198](#) and described in [Table 3-222](#).

Return to the [Summary Table](#).

Local Shared RAM TEST Register

**Figure 3-198. LSxTEST 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
TEST_LS7		TEST_LS6		TEST_LS5		TEST_LS4	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
TEST_LS3		TEST_LS2		TEST_LS1		TEST_LS0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 3-222. LSxTEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	TEST_LS7	R/W	0h	Selects the defferent modes for LS7 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
13-12	TEST_LS6	R/W	0h	Selects the defferent modes for LS6 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
11-10	TEST_LS5	R/W	0h	Selects the defferent modes for LS5 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
9-8	TEST_LS4	R/W	0h	Selects the defferent modes for LS4 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn

**Table 3-222. LSxTEST Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	TEST_LS3	R/W	0h	Selects the defferent modes for LS3 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
5-4	TEST_LS2	R/W	0h	Selects the defferent modes for LS2 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
3-2	TEST_LS1	R/W	0h	Selects the defferent modes for LS1 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
1-0	TEST_LS0	R/W	0h	Selects the defferent modes for LS0 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn

**3.14.15.14 LSxINIT Register (Offset = 32h) [reset = 0h]**

LSxINIT is shown in [Figure 3-199](#) and described in [Table 3-223](#).

Return to the [Summary Table](#).

Local Shared RAM Init Register

**Figure 3-199. LSxINIT 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
INIT_LS7	INIT_LS6	INIT_LS5	INIT_LS4	INIT_LS3	INIT_LS2	INIT_LS1	INIT_LS0
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-223. LSxINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	INIT_LS7	R-0/W1S	0h	RAM Initialization control for LS7 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
6	INIT_LS6	R-0/W1S	0h	RAM Initialization control for LS6 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
5	INIT_LS5	R-0/W1S	0h	RAM Initialization control for LS5 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
4	INIT_LS4	R-0/W1S	0h	RAM Initialization control for LS4 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
3	INIT_LS3	R-0/W1S	0h	RAM Initialization control for LS3 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
2	INIT_LS2	R-0/W1S	0h	RAM Initialization control for LS2 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn

**Table 3-223. LSxINIT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INIT_LS1	R-0/W1S	0h	RAM Initialization control for LS1 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
0	INIT_LS0	R-0/W1S	0h	RAM Initialization control for LS0 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn

**3.14.15.15 LSxINITDONE Register (Offset = 34h) [reset = 0h]**

LSxINITDONE is shown in [Figure 3-200](#) and described in [Table 3-224](#).

Return to the [Summary Table](#).

Local Shared RAM InitDone Status Register

**Figure 3-200. LSxINITDONE 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
INITDONE_LS7	INITDONE_LS6	INITDONE_LS5	INITDONE_LS4	INITDONE_LS3	INITDONE_LS2	INITDONE_LS1	INITDONE_LS0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-224. LSxINITDONE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	INITDONE_LS7	R	0h	RAM Initialization status for LS7 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
6	INITDONE_LS6	R	0h	RAM Initialization status for LS6 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
5	INITDONE_LS5	R	0h	RAM Initialization status for LS5 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
4	INITDONE_LS4	R	0h	RAM Initialization status for LS4 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
3	INITDONE_LS3	R	0h	RAM Initialization status for LS3 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
2	INITDONE_LS2	R	0h	RAM Initialization status for LS2 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn



**Table 3-224. LSxINITDONE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INITDONE_LS1	R	0h	RAM Initialization status for LS1 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
0	INITDONE_LS0	R	0h	RAM Initialization status for LS0 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn

### 3.14.15.16 GSxLOCK Register (Offset = 40h) [reset = 0h]

GSxLOCK is shown in [Figure 3-201](#) and described in [Table 3-225](#).

Return to the [Summary Table](#).

Global Shared RAM Config Lock Register

**Figure 3-201. GSxLOCK 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	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	LOCK_GS3	LOCK_GS2	LOCK_GS1	LOCK_GS0
				R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-225. GSxLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	LOCK_GS3	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for GS3 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked. Reset type: SYSRSn
2	LOCK_GS2	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for GS2 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked. Reset type: SYSRSn
1	LOCK_GS1	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for GS1 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked. Reset type: SYSRSn

**Table 3-225. GSxLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	LOCK_GS0	R/W	0h	<p>Locks the write to access protection, master select, initialization control and test register fields for GS0 RAM:</p> <p>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed.</p> <p>1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked.</p> <p>Reset type: SYSRSn</p>

**3.14.15.17 GSxCOMMIT Register (Offset = 42h) [reset = 0h]**

 GSxCOMMIT is shown in [Figure 3-202](#) and described in [Table 3-226](#).

 Return to the [Summary Table](#).

Global Shared RAM Config Lock Commit Register

**Figure 3-202. GSxCOMMIT 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	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	COMMIT_GS3	COMMIT_GS2	COMMIT_GS1	COMMIT_GS0
				R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 3-226. GSxCOMMIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	RESERVED	R/WOnce	0h	Reserved
14	RESERVED	R/WOnce	0h	Reserved
13	RESERVED	R/WOnce	0h	Reserved
12	RESERVED	R/WOnce	0h	Reserved
11	RESERVED	R/WOnce	0h	Reserved
10	RESERVED	R/WOnce	0h	Reserved
9	RESERVED	R/WOnce	0h	Reserved
8	RESERVED	R/WOnce	0h	Reserved
7	RESERVED	R/WOnce	0h	Reserved
6	RESERVED	R/WOnce	0h	Reserved
5	RESERVED	R/WOnce	0h	Reserved
4	RESERVED	R/WOnce	0h	Reserved
3	COMMIT_GS3	R/WOnce	0h	Permanently Locks the write to access protection, master select, initialization control and test register fields for GS3 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked. Reset type: SYSRSn
2	COMMIT_GS2	R/WOnce	0h	Permanently Locks the write to access protection, master select, initialization control and test register fields for GS2 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked. Reset type: SYSRSn

**Table 3-226. GSxCOMMIT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	COMMIT_GS1	R/WOnce	0h	Permanently Locks the write to access protection, master select, initialization control and test register fields for GS1 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked. Reset type: SYSRSn
0	COMMIT_GS0	R/WOnce	0h	Permanently Locks the write to access protection, master select, initialization control and test register fields for GS0 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked. Reset type: SYSRSn

**3.14.15.18 GSxACCPROT0 Register (Offset = 48h) [reset = 0h]**

 GSxACCPROT0 is shown in [Figure 3-203](#) and described in [Table 3-227](#).

 Return to the [Summary Table](#).

Global Shared RAM Config Register 0

**Figure 3-203. GSxACCPROT0 Register**

31	30	29	28	27	26	25	24
RESERVED					DMAWRPROT_GS3	CPUWRPROT_GS3	FETCHPROT_GS3
R-0h					R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED					DMAWRPROT_GS2	CPUWRPROT_GS2	FETCHPROT_GS2
R-0h					R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED					DMAWRPROT_GS1	CPUWRPROT_GS1	FETCHPROT_GS1
R-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED					DMAWRPROT_GS0	CPUWRPROT_GS0	FETCHPROT_GS0
R-0h					R/W-0h	R/W-0h	R/W-0h

**Table 3-227. GSxACCPROT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26	DMAWRPROT_GS3	R/W	0h	DMA WR Protection For GS3 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked. Reset type: SYSRSn
25	CPUWRPROT_GS3	R/W	0h	CPU WR Protection For GS3 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
24	FETCHPROT_GS3	R/W	0h	Fetch Protection For GS3 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
23-19	RESERVED	R	0h	Reserved
18	DMAWRPROT_GS2	R/W	0h	DMA WR Protection For GS2 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked. Reset type: SYSRSn
17	CPUWRPROT_GS2	R/W	0h	CPU WR Protection For GS2 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn

**Table 3-227. GSxACCPROT0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	FETCHPROT_GS2	R/W	0h	Fetch Protection For GS2 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
15-11	RESERVED	R	0h	Reserved
10	DMAWRPROT_GS1	R/W	0h	DMA WR Protection For GS1 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked. Reset type: SYSRSn
9	CPUWRPROT_GS1	R/W	0h	CPU WR Protection For GS1 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
8	FETCHPROT_GS1	R/W	0h	Fetch Protection For GS1 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
7-3	RESERVED	R	0h	Reserved
2	DMAWRPROT_GS0	R/W	0h	DMA WR Protection For GS0 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked. Reset type: SYSRSn
1	CPUWRPROT_GS0	R/W	0h	CPU WR Protection For GS0 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
0	FETCHPROT_GS0	R/W	0h	Fetch Protection For GS0 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn

**3.14.15.19 GSxACCPROT1 Register (Offset = 4Ah) [reset = 0h]**

GSxACCPROT1 is shown in [Figure 3-204](#) and described in [Table 3-228](#).

Return to the [Summary Table](#).

Global Shared RAM Config Register 1

**Figure 3-204. GSxACCPROT1 Register**

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

**Table 3-228. GSxACCPROT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23-19	RESERVED	R	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-11	RESERVED	R	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7-3	RESERVED	R	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved



**3.14.15.20 GSxACCPROT2 Register (Offset = 4Ch) [reset = 0h]**

GSxACCPROT2 is shown in [Figure 3-205](#) and described in [Table 3-229](#).

Return to the [Summary Table](#).

Global Shared RAM Config Register 2

**Figure 3-205. GSxACCPROT2 Register**

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

**Table 3-229. GSxACCPROT2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23-19	RESERVED	R	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-11	RESERVED	R	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7-3	RESERVED	R	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

**3.14.15.21 GSxACCPROT3 Register (Offset = 4Eh) [reset = 0h]**

GSxACCPROT3 is shown in [Figure 3-206](#) and described in [Table 3-230](#).

Return to the [Summary Table](#).

Global Shared RAM Config Register 3

**Figure 3-206. GSxACCPROT3 Register**

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

**Table 3-230. GSxACCPROT3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23-19	RESERVED	R	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-11	RESERVED	R	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7-3	RESERVED	R	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

**3.14.15.22 GSxTEST Register (Offset = 50h) [reset = 0h]**

GSxTEST is shown in [Figure 3-207](#) and described in [Table 3-231](#).

Return to the [Summary Table](#).

Global Shared RAM TEST Register

**Figure 3-207. GSxTEST Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED		RESERVED		RESERVED	
23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED	
15	14	13	12	11	10	9	8
RESERVED		RESERVED		RESERVED		RESERVED	
7	6	5	4	3	2	1	0
TEST_GS3		TEST_GS2		TEST_GS1		TEST_GS0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 3-231. GSxTEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15-14	RESERVED	R/W	0h	Reserved
13-12	RESERVED	R/W	0h	Reserved
11-10	RESERVED	R/W	0h	Reserved
9-8	RESERVED	R/W	0h	Reserved
7-6	TEST_GS3	R/W	0h	Selects the defferent modes for GS3 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
5-4	TEST_GS2	R/W	0h	Selects the defferent modes for GS2 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn

**Table 3-231. GSxTEST Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	TEST_GS1	R/W	0h	Selects the defferent modes for GS1 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
1-0	TEST_GS0	R/W	0h	Selects the defferent modes for GS0 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn

**3.14.15.23 GSxINIT Register (Offset = 52h) [reset = 0h]**

 GSxINIT is shown in [Figure 3-208](#) and described in [Table 3-232](#).

 Return to the [Summary Table](#).

Global Shared RAM Init Register

**Figure 3-208. GSxINIT 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	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	INIT_GS3	INIT_GS2	INIT_GS1	INIT_GS0
				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-232. GSxINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	RESERVED	R-0/W1S	0h	Reserved
14	RESERVED	R-0/W1S	0h	Reserved
13	RESERVED	R-0/W1S	0h	Reserved
12	RESERVED	R-0/W1S	0h	Reserved
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	RESERVED	R-0/W1S	0h	Reserved
6	RESERVED	R-0/W1S	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	INIT_GS3	R-0/W1S	0h	RAM Initialization control for GS3 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
2	INIT_GS2	R-0/W1S	0h	RAM Initialization control for GS2 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
1	INIT_GS1	R-0/W1S	0h	RAM Initialization control for GS1 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn

**Table 3-232. GSxINIT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	INIT_GS0	R-0/W1S	0h	RAM Initialization control for GS0 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn

**3.14.15.24 GSxINITDONE Register (Offset = 54h) [reset = 0h]**

 GSxINITDONE is shown in [Figure 3-209](#) and described in [Table 3-233](#).

 Return to the [Summary Table](#).

Global Shared RAM InitDone Status Register

**Figure 3-209. GSxINITDONE 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	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	INITDONE_GS 3	INITDONE_GS 2	INITDONE_GS 1	INITDONE_GS 0
R-0h				R-0h		R-0h	

**Table 3-233. GSxINITDONE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	RESERVED	R	0h	Reserved
14	RESERVED	R	0h	Reserved
13	RESERVED	R	0h	Reserved
12	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	INITDONE_GS3	R	0h	RAM Initialization status for GS3 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
2	INITDONE_GS2	R	0h	RAM Initialization status for GS2 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
1	INITDONE_GS1	R	0h	RAM Initialization status for GS1 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn

**Table 3-233. GSxINITDONE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	INITDONE_GS0	R	0h	RAM Initialization status for GS0 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn



**3.14.15.25 MSGxLOCK Register (Offset = 60h) [reset = 0h]**

MSGxLOCK is shown in [Figure 3-210](#) and described in [Table 3-234](#).

Return to the [Summary Table](#).

Message RAM Config Lock Register

**Figure 3-210. MSGxLOCK 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			RESERVED	RESERVED	LOCK_CLA1T OCPU	LOCK_CPUTO CLA1	RESERVED
R-0h					R/W-0h	R/W-0h	

**Table 3-234. MSGxLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	LOCK_CLA1TOCPU	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for CLA1TOCPU RAM: 0: Write to TEST, INIT fields are allowed. 1: Write to TEST, INIT fields are blocked. Reset type: SYSRSn
1	LOCK_CPUTOCLA1	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for CPUTOCLA1 RAM: 0: Write to TEST, INIT fields are allowed. 1: Write to TEST, INIT fields are blocked. Reset type: SYSRSn
0	RESERVED	R/W	0h	Reserved

**3.14.15.26 MSGxCOMMIT Register (Offset = 62h) [reset = 0h]**

MSGxCOMMIT is shown in [Figure 3-211](#) and described in [Table 3-235](#).

Return to the [Summary Table](#).

Message RAM Config Lock Commit Register

**Figure 3-211. MSGxCOMMIT 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			RESERVED	RESERVED	COMMIT_CLA1 TOCPU	COMMIT_CPU TOCLA1	RESERVED
R-0h					R/WOnce-0h	R/WOnce-0h	

**Table 3-235. MSGxCOMMIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	RESERVED	R/WOnce	0h	Reserved
3	RESERVED	R/WOnce	0h	Reserved
2	COMMIT_CLA1TOCPU	R/WOnce	0h	Locks the write to access protection, master select, initialization control and test register fields for CLA1TOCPU RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked. Reset type: SYSRSn
1	COMMIT_CPUTOCLA1	R/WOnce	0h	Locks the write to access protection, master select, initialization control and test register fields for CPUTOCLA1 RAM: 0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed. 1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked. Reset type: SYSRSn
0	RESERVED	R/WOnce	0h	Reserved

**3.14.15.27 MSGxTEST Register (Offset = 70h) [reset = 0h]**

MSGxTEST is shown in [Figure 3-212](#) and described in [Table 3-236](#).

Return to the [Summary Table](#).

Message RAM TEST Register

**Figure 3-212. MSGxTEST 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						RESERVED	
R-0h							
7	6	5	4	3	2	1	0
RESERVED		TEST_CLA1TOCPU		TEST_CPUTOCLA1		RESERVED	
R/W-0h				R/W-0h			

**Table 3-236. MSGxTEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-10	RESERVED	R	0h	Reserved
9-8	RESERVED	R/W	0h	Reserved
7-6	RESERVED	R/W	0h	Reserved
5-4	TEST_CLA1TOCPU	R/W	0h	Selects the defferent modes for CLA1TOCPU MSG RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
3-2	TEST_CPUTOCLA1	R/W	0h	Selects the defferent modes for CPUTOCLA1 MSG RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Functional Mode. Reset type: SYSRSn
1-0	RESERVED	R/W	0h	Reserved

**3.14.15.28 MSGxINIT Register (Offset = 72h) [reset = 0h]**

MSGxINIT is shown in [Figure 3-213](#) and described in [Table 3-237](#).

Return to the [Summary Table](#).

Message RAM Init Register

**Figure 3-213. MSGxINIT 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			RESERVED	RESERVED	INIT_CLA1TOC PU	INIT_CPUTOC LA1	RESERVED
R-0h					R-0/W1S-0h	R-0/W1S-0h	

**Table 3-237. MSGxINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	INIT_CLA1TOCPU	R-0/W1S	0h	RAM Initialization control for CLA1TOCPU MSG RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
1	INIT_CPUTOCLA1	R-0/W1S	0h	RAM Initialization control for CPUTOCLA1 MSG RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
0	RESERVED	R-0/W1S	0h	Reserved

**3.14.15.29 MSGxINITDONE Register (Offset = 74h) [reset = 0h]**

MSGxINITDONE is shown in [Figure 3-214](#) and described in [Table 3-238](#).

Return to the [Summary Table](#).

Message RAM InitDone Status Register

**Figure 3-214. MSGxINITDONE 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			RESERVED	RESERVED	INITDONE_CL A1TOCPU	INITDONE_CP UTOCLA1	RESERVED
R-0h					R-0h	R-0h	

**Table 3-238. MSGxINITDONE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	INITDONE_CLA1TOCPU	R	0h	RAM Initialization status for CLA1TOCPU MSG RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
1	INITDONE_CPUTOCLA1	R	0h	RAM Initialization status for CPUTOCLA1 MSG RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
0	RESERVED	R	0h	Reserved

### 3.14.16 MEMORY\_ERROR\_REGS Registers

Table 3-239 lists the MEMORY\_ERROR\_REGS registers. All register offset addresses not listed in Table 3-239 should be considered as reserved locations and the register contents should not be modified.

**Table 3-239. MEMORY\_ERROR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	UCERRFLG	Uncorrectable Error Flag Register		<a href="#">Go</a>
2h	UCERRSET	Uncorrectable Error Flag Set Register	EALLOW	<a href="#">Go</a>
4h	UCERRCLR	Uncorrectable Error Flag Clear Register	EALLOW	<a href="#">Go</a>
6h	UCCPUREADDR	Uncorrectable CPU Read Error Address		<a href="#">Go</a>
8h	UCDMAREADDR	Uncorrectable DMA Read Error Address		<a href="#">Go</a>
Ah	UCCLA1READDR	Uncorrectable CLA1 Read Error Address		<a href="#">Go</a>
20h	CERRFLG	Correctable Error Flag Register		<a href="#">Go</a>
22h	CERRSET	Correctable Error Flag Set Register	EALLOW	<a href="#">Go</a>
24h	CERRCLR	Correctable Error Flag Clear Register	EALLOW	<a href="#">Go</a>
26h	CCPUREADDR	Correctable CPU Read Error Address		<a href="#">Go</a>
2Eh	CERRCNT	Correctable Error Count Register		<a href="#">Go</a>
30h	CERRTHRES	Correctable Error Threshold Value Register	EALLOW	<a href="#">Go</a>
32h	CEINTFLG	Correctable Error Interrupt Flag Status Register		<a href="#">Go</a>
34h	CEINTCLR	Correctable Error Interrupt Flag Clear Register	EALLOW	<a href="#">Go</a>
36h	CEINTSET	Correctable Error Interrupt Flag Set Register	EALLOW	<a href="#">Go</a>
38h	CEINTEN	Correctable Error Interrupt Enable Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-240 shows the codes that are used for access types in this section.

**Table 3-240. MEMORY\_ERROR\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.16.1 UCERRFLG Register (Offset = 0h) [reset = 0h]**

UCERRFLG is shown in [Figure 3-215](#) and described in [Table 3-241](#).

Return to the [Summary Table](#).

Uncorrectable Error Flag Register

**Figure 3-215. UCERRFLG 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				RESERVED	CLA1RDERR	DMARDERR	CPURDERR
R-0h					R-0h	R-0h	R-0h

**Table 3-241. UCERRFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	CLA1RDERR	R	0h	CLA1 Uncorrectable Read Error Flag 0: No Error. 1: Uncorrectable error occurred during CLA1 read. Reset type: SYSRSn
1	DMARDERR	R	0h	DMA Uncorrectable Read Error Flag 0: No Error. 1: Uncorrectable error occurred during DMA read. Reset type: SYSRSn
0	CPURDERR	R	0h	CPU Uncorrectable Read Error Flag 0: No Error. 1: Uncorrectable error occurred during CPU read. Reset type: SYSRSn

**3.14.16.2 UCERRSET Register (Offset = 2h) [reset = 0h]**

UCERRSET is shown in [Figure 3-216](#) and described in [Table 3-242](#).

Return to the [Summary Table](#).

Uncorrectable Error Flag Set Register

**Figure 3-216. UCERRSET 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				RESERVED	CLA1RDERR	DMARDERR	CPURDERR
R-0h					R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-242. UCERRSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	CLA1RDERR	R-0/W1S	0h	0: No action. 1: CLA1 Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled.. Reset type: SYSRSn
1	DMARDERR	R-0/W1S	0h	0: No action. 1: DMA Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled.. Reset type: SYSRSn
0	CPURDERR	R-0/W1S	0h	0: No action. 1: CPU Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled.. Reset type: SYSRSn



### 3.14.16.3 UCERRCLR Register (Offset = 4h) [reset = 0h]

UCERRCLR is shown in [Figure 3-217](#) and described in [Table 3-243](#).

Return to the [Summary Table](#).

Uncorrectable Error Flag Clear Register

**Figure 3-217. UCERRCLR 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				RESERVED	CLA1RDERR	DMARDERR	CPURDERR
R-0h					R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-243. UCERRCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	CLA1RDERR	R-0/W1S	0h	0: No action. 1: CLA1 Read Error Flag in UCERRFLG register will be cleared. Reset type: SYSRSn
1	DMARDERR	R-0/W1S	0h	0: No action. 1: DMA Read Error Flag in UCERRFLG register will be cleared . Reset type: SYSRSn
0	CPURDERR	R-0/W1S	0h	0: No action. 1: CPU Read Error Flag in UCERRFLG register will be cleared. Reset type: SYSRSn

### 3.14.16.4 UCCPUREADDR Register (Offset = 6h) [reset = 0h]

UCCPUREADDR is shown in [Figure 3-218](#) and described in [Table 3-244](#).

Return to the [Summary Table](#).

Uncorrectable CPU Read Error Address

**Figure 3-218. UCCPUREADDR 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
UCCPUREADDR																															
R-0h																															

**Table 3-244. UCCPUREADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	UCCPUREADDR	R	0h	This register captures the address location for which CPU read/fetch access resulted in uncorrectable ECC/Parity error. Reset type: SYSRSn

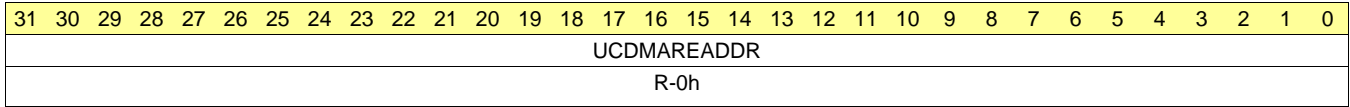
**3.14.16.5 UCDMAREADDR Register (Offset = 8h) [reset = 0h]**

UCDMAREADDR is shown in [Figure 3-219](#) and described in [Table 3-245](#).

Return to the [Summary Table](#).

Uncorrectable DMA Read Error Address

**Figure 3-219. UCDMAREADDR Register**



**Table 3-245. UCDMAREADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	UCDMAREADDR	R	0h	This register captures the address location for which DMA read access resulted in uncorrectable Parity error. Reset type: SYSRSn

### 3.14.16.6 UCCLA1READDR Register (Offset = Ah) [reset = 0h]

UCCLA1READDR is shown in [Figure 3-220](#) and described in [Table 3-246](#).

Return to the [Summary Table](#).

Uncorrectable CLA1 Read Error Address

**Figure 3-220. UCCLA1READDR 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
UCCLA1READDR																															
R-0h																															

**Table 3-246. UCCLA1READDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	UCCLA1READDR	R	0h	This register captures the address location for which CLA1 read/fetch access resulted in uncorrectable Parity error. Reset type: SYSRSn

**3.14.16.7 CERRFLG Register (Offset = 20h) [reset = 0h]**

CERRFLG is shown in [Figure 3-221](#) and described in [Table 3-247](#).

Return to the [Summary Table](#).

Correctable Error Flag Register

**Figure 3-221. CERRFLG 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				RESERVED	CLA1RDERR	DMARDERR	CPURDERR
R-0h					R-0h	R-0h	R-0h

**Table 3-247. CERRFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	CLA1RDERR	R	0h	CLA1 Correctable Read Error Flag 0: No Error. 1: Correctable error occurred during CLA1 read. Reset type: SYSRSn
1	DMARDERR	R	0h	DMA Correctable Read Error Flag 0: No Error. 1: Correctable error occurred during DMA read. Reset type: SYSRSn
0	CPURDERR	R	0h	CPU Correctable Read Error Flag 0: No Error. 1: Correctable error occurred during CPU read. Reset type: SYSRSn

**3.14.16.8 CERRSET Register (Offset = 22h) [reset = 0h]**

CERRSET is shown in [Figure 3-222](#) and described in [Table 3-248](#).

Return to the [Summary Table](#).

Correctable Error Flag Set Register

**Figure 3-222. CERRSET 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				RESERVED	CLA1RDERR	DMARDERR	CPURDERR
R-0h					R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-248. CERRSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	CLA1RDERR	R-0/W1S	0h	0: No action. 1: CLA1 Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled.. Reset type: SYSRSn
1	DMARDERR	R-0/W1S	0h	0: No action. 1: DMA Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled.. Reset type: SYSRSn
0	CPURDERR	R-0/W1S	0h	0: No action. 1: CPU Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled.. Reset type: SYSRSn

**3.14.16.9 CERRCLR Register (Offset = 24h) [reset = 0h]**

CERRCLR is shown in [Figure 3-223](#) and described in [Table 3-249](#).

Return to the [Summary Table](#).

Correctable Error Flag Clear Register

**Figure 3-223. CERRCLR 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				RESERVED	CLA1RDERR	DMARDERR	CPURDERR
R-0h					R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-249. CERRCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	CLA1RDERR	R-0/W1S	0h	0: No action. 1: CLA1 Read Error Flag in CERRFLG register will be cleared. Reset type: SYSRSn
1	DMARDERR	R-0/W1S	0h	0: No action. 1: DMA Read Error Flag in CERRFLG register will be cleared . Reset type: SYSRSn
0	CPURDERR	R-0/W1S	0h	0: No action. 1: CPU Read Error Flag in CERRFLG register will be cleared. Reset type: SYSRSn

**3.14.16.10 CCPUREADDR Register (Offset = 26h) [reset = 0h]**

CCPUREADDR is shown in [Figure 3-224](#) and described in [Table 3-250](#).

Return to the [Summary Table](#).

Correctable CPU Read Error Address

**Figure 3-224. CCPUREADDR 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
CCPUREADDR																															
R-0h																															

**Table 3-250. CCPUREADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CCPUREADDR	R	0h	This register captures the address location for which CPU read/fetch access resulted in correctable ECC error. Reset type: SYSRSn



**3.14.16.11 CERRCNT Register (Offset = 2Eh) [reset = 0h]**

CERRCNT is shown in [Figure 3-225](#) and described in [Table 3-251](#).

Return to the [Summary Table](#).

Correctable Error Count Register

**Figure 3-225. CERRCNT 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
CERRCNT																															
R-0h																															

**Table 3-251. CERRCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CERRCNT	R	0h	This register holds the count of how many times correctable error occurred. Reset type: SYSRSn

**3.14.16.12 CERRTHRES Register (Offset = 30h) [reset = 0h]**

CERRTHRES is shown in [Figure 3-226](#) and described in [Table 3-252](#).

Return to the [Summary Table](#).

Correctable Error Threshold Value Register

**Figure 3-226. CERRTHRES 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
CERRTHRES																															
R/W-0h																															

**Table 3-252. CERRTHRES Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CERRTHRES	R/W	0h	When value in CERRCNT register is greater than value configured in this register, correctable interrupt gets generated, if enabled. Reset type: SYSRSn

**3.14.16.13 CEINTFLG Register (Offset = 32h) [reset = 0h]**

CEINTFLG is shown in [Figure 3-227](#) and described in [Table 3-253](#).

Return to the [Summary Table](#).

Correctable Error Interrupt Flag Status Register

**Figure 3-227. CEINTFLG 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							CEINTFLAG
R-0h							R-0h

**Table 3-253. CEINTFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	CEINTFLAG	R	0h	Total corrected error count exceeded threshold Flag 0: Total correctable errors < Threshold value configured in CERRTHRES register. 1: Total correctable errors >= Threshold value configured in CERRTHRES register. Reset type: SYSRSn

**3.14.16.14 CEINTCLR Register (Offset = 34h) [reset = 0h]**

CEINTCLR is shown in [Figure 3-228](#) and described in [Table 3-254](#).

Return to the [Summary Table](#).

Correctable Error Interrupt Flag Clear Register

**Figure 3-228. CEINTCLR 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							CEINTCLR
R-0h							R-0/W1S-0h

**Table 3-254. CEINTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	CEINTCLR	R-0/W1S	0h	0: No action. 1: Total corrected error count exceeded flag in CEINTFLG register will be cleared. Reset type: SYSRSn

**3.14.16.15 CEINTSET Register (Offset = 36h) [reset = 0h]**

CEINTSET is shown in [Figure 3-229](#) and described in [Table 3-255](#).

Return to the [Summary Table](#).

Correctable Error Interrupt Flag Set Register

**Figure 3-229. CEINTSET 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							CEINTSET
R-0h							R-0/W1S-0h

**Table 3-255. CEINTSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	CEINTSET	R-0/W1S	0h	0: No action. 1: Total corrected error count exceeded flag in CEINTFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn

**3.14.16.16 CEINTEN Register (Offset = 38h) [reset = 0h]**

CEINTEN is shown in [Figure 3-230](#) and described in [Table 3-256](#).

Return to the [Summary Table](#).

Correctable Error Interrupt Enable Register

**Figure 3-230. CEINTEN 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							CEINTEN
R-0h							R/W-0h

**Table 3-256. CEINTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	CEINTEN	R/W	0h	0: Correctable Error Interrupt is disabled. 1: Correctable Error Interrupt is enabled. Reset type: SYSRSn

### 3.14.17 NMI\_INTRUPT\_REGS Registers

Table 3-257 lists the NMI\_INTRUPT\_REGS registers. All register offset addresses not listed in Table 3-257 should be considered as reserved locations and the register contents should not be modified.

**Table 3-257. NMI\_INTRUPT\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	NMICFG	NMI Configuration Register	EALLOW	<a href="#">Go</a>
1h	NMIFLG	NMI Flag Register (SYSRSn Clear)		<a href="#">Go</a>
2h	NMIFLGCLR	NMI Flag Clear Register	EALLOW	<a href="#">Go</a>
3h	NMIFLGFRC	NMI Flag Force Register	EALLOW	<a href="#">Go</a>
4h	NMIWDCNT	NMI Watchdog Counter Register		<a href="#">Go</a>
5h	NMIWDPRD	NMI Watchdog Period Register	EALLOW	<a href="#">Go</a>
6h	NMISHDFLG	NMI Shadow Flag Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-258 shows the codes that are used for access types in this section.

**Table 3-258. NMI\_INTRUPT\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R -0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W 1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.17.1 NMICFG Register (Offset = 0h) [reset = 0h]

NMICFG is shown in [Figure 3-231](#) and described in [Table 3-259](#).

Return to the [Summary Table](#).

NMI Configuration Register

**Figure 3-231. NMICFG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED							NMIE
R-0-0h							R/W1S-0h

**Table 3-259. NMICFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R-0	0h	Reserved
0	NMIE	R/W1S	0h	Global NMI Enable  This bit indicates that the NMI module has been enabled, which allows system error conditions to trigger an NMI to the CPU. The boot ROM sets this bit at start-up. It can only be cleared by a system reset.  Reset type: SYSRStn



### 3.14.17.2 NMIFLG Register (Offset = 1h) [reset = 0h]

NMIFLG is shown in [Figure 3-232](#) and described in [Table 3-260](#).

Return to the [Summary Table](#).

NMI Flag Register (SYSRSn Clear)

**Figure 3-232. NMIFLG Register**

15	14	13	12	11	10	9	8
RESERVED		SWERR	RESERVED	RESERVED	RESERVED	RESERVED	CLBNMI
R-0-0h		R-0h					R-0h
7	6	5	4	3	2	1	0
RESERVED	PIEVECTERR	RESERVED	RESERVED	FLUNCERR	RAMUNCERR	CLOCKFAIL	NMIINT
R-0h				R-0h	R-0h	R-0h	R-0h

**Table 3-260. NMIFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R-0	0h	Reserved
13	SWERR	R	0h	Software-Forced NMI Flag  This bit can only be set by writing to the corresponding bit in the NMIFLGFRC register. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register. No further NMIs are generated until this flag is cleared.  Reset type: SYSRSn 0h (R/W) = No software NMI forced 1h (R/W) = Software NMI forced
12	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	CLBNMI	R	0h	CLB NMI Flag  This bit indicates whether an NMI was generated by the CLB. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register.  Reset type: SYSRSn 0h (R/W) = No CLB NMI generated 1h (R/W) = CLB NMI generated
7	RESERVED	R	0h	Reserved
6	PIEVECTERR	R	0h	PIE Vector Fetch Error NMI Flag  This bit indicates whether a mismatch was detected during an interrupt vector fetch. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register.  Reset type: SYSRSn 0h (R/W) = No vector fetch error detected 1h (R/W) = Vector fetch error detected
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved

**Table 3-260. NMIFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	FLUNCERR	R	0h	Flash Uncorrectable Error NMI Flag  This bit indicates whether an uncorrectable ECC error occurred during a flash access. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register. Reset type: SYSRSn 0h (R/W) = No uncorrectable error detected 1h (R/W) = Uncorrectable error detected
2	RAMUNCERR	R	0h	RAM Uncorrectable Error NMI Flag  This bit indicates whether an uncorrectable ECC error has occurred during a RAM access by any master. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register. Reset type: SYSRSn 0h (R/W) = No uncorrectable error detected 1h (R/W) = Uncorrectable error detected
1	CLOCKFAIL	R	0h	Clock Fail NMI Flag  This bit indicates whether a clock fail condition has been detected. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register. Reset type: SYSRSn 0h (R/W) = No clock fail detected 1h (R/W) = Clock fail detected
0	NMIINT	R	0h	Global NMI Flag  This bit indicates whether an NMI has been generated. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register. No further NMIs are generated until this flag is cleared. Reset type: SYSRSn 0h (R/W) = No NMI generated 1h (R/W) = NMI generated

### 3.14.17.3 NMIFLGCLR Register (Offset = 2h) [reset = 0h]

NMIFLGCLR is shown in [Figure 3-233](#) and described in [Table 3-261](#).

Return to the [Summary Table](#).

Writing a 1 to one of these bits clears the corresponding bit in the NMIFLG register. Writes of 0 are ignored, and these bits always read 0. If an NMI arrives on the same cycle that this register is written, the NMI is latched. All other NMI flags must be cleared before the NMIINT flag is cleared, otherwise NMIINT will be set again.

**Figure 3-233. NMIFLGCLR Register**

15		14		13		12		11		10		9		8	
RESERVED				SWERR		RESERVED		RESERVED		RESERVED		RESERVED		CLBNMI	
R-0-0h				R-0/W1S-0h								R-0/W1S-0h			
7		6		5		4		3		2		1		0	
RESERVED		PIEVECTERR		RESERVED		RESERVED		FLUNCERR		RAMUNCERR		CLOCKFAIL		NMIINT	
R-0/W1S-0h								R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h	

**Table 3-261. NMIFLGCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R-0	0h	Reserved
13	SWERR	R-0/W1S	0h	Clear the SWERR flag. Reset type: SYSRSn
12	RESERVED	R-0/W1S	0h	Reserved
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	CLBNMI	R-0/W1S	0h	Clear the CLBNMI flag. Reset type: SYSRSn
7	RESERVED	R-0/W1S	0h	Reserved
6	PIEVECTERR	R-0/W1S	0h	Clear the PIEVECTERR flag. Reset type: SYSRSn
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	FLUNCERR	R-0/W1S	0h	Clear the FLUNCERR flag. Reset type: SYSRSn
2	RAMUNCERR	R-0/W1S	0h	Clear the RAMUNCERR flag. Reset type: SYSRSn
1	CLOCKFAIL	R-0/W1S	0h	Clear the CLOCKFAIL flag. Reset type: SYSRSn
0	NMIINT	R-0/W1S	0h	Clear the NMIINT flag. This flag should only be cleared after all other active flags have been cleared. Reset type: SYSRSn

### 3.14.17.4 NMIFLGFRFC Register (Offset = 3h) [reset = 0h]

NMIFLGFRFC is shown in [Figure 3-234](#) and described in [Table 3-262](#).

Return to the [Summary Table](#).

Writing a 1 to one of these bits sets the corresponding bit in the NMIFLG register. Writes of 0 are ignored, and these bits always read 0. This register can be used to test the NMI functionality.

**Figure 3-234. NMIFLGFRFC Register**

15	14	13	12	11	10	9	8
RESERVED		SWERR	RESERVED	RESERVED	RESERVED	RESERVED	CLBNMI
R-0-0h		R-0/W1S-0h				R-0/W1S-0h	
7	6	5	4	3	2	1	0
RESERVED	PIEVECTERR	RESERVED	RESERVED	FLUNCERR	RAMUNCERR	CLOCKFAIL	RESERVED
R-0/W1S-0h				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0-0h

**Table 3-262. NMIFLGFRFC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R-0	0h	Reserved
13	SWERR	R-0/W1S	0h	Set the SWERR flag. Reset type: SYSRSn
12	RESERVED	R-0/W1S	0h	Reserved
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	CLBNMI	R-0/W1S	0h	Set the CLBNMI flag. Reset type: SYSRSn
7	RESERVED	R-0/W1S	0h	Reserved
6	PIEVECTERR	R-0/W1S	0h	Set the PIEVECTERR flag. Reset type: SYSRSn
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	FLUNCERR	R-0/W1S	0h	Set the FLUNCERR flag. Reset type: SYSRSn
2	RAMUNCERR	R-0/W1S	0h	Set the RAMUNCERR flag. Reset type: SYSRSn
1	CLOCKFAIL	R-0/W1S	0h	Set the CLOCKFAIL flag. Reset type: SYSRSn
0	RESERVED	R-0	0h	Reserved

### 3.14.17.5 NMIWDCNT Register (Offset = 4h) [reset = 0h]

NMIWDCNT is shown in [Figure 3-235](#) and described in [Table 3-263](#).

Return to the [Summary Table](#).

NMI Watchdog Counter Register

**Figure 3-235. NMIWDCNT Register**

15	14	13	12	11	10	9	8
NMIWDCNT							
R-0h							
7	6	5	4	3	2	1	0
NMIWDCNT							
R-0h							

**Table 3-263. NMIWDCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	NMIWDCNT	R	0h	<p>NMI Watchdog Counter</p> <p>This 16-bit counter increments once per SYSCLK cycle whenever any of the NMIFLG bits are set. If the counter reaches the period value in the NMIWDPRD register, the NMI module generates a reset (NMIWDRS). After this reset, the counter resets to zero and stops counting.</p> <p>If all NMI flags are cleared, the counter will reset to zero and stop counting until another NMI flag is set.</p> <p>Reset type: SYSRSn</p>

### 3.14.17.6 NMIWDPRD Register (Offset = 5h) [reset = FFFFh]

NMIWDPRD is shown in [Figure 3-236](#) and described in [Table 3-264](#).

Return to the [Summary Table](#).

NMI Watchdog Period Register

**Figure 3-236. NMIWDPRD Register**

15	14	13	12	11	10	9	8
NMIWDPRD							
R/W-FFFFh							
7	6	5	4	3	2	1	0
NMIWDPRD							
R/W-FFFFh							

**Table 3-264. NMIWDPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	NMIWDPRD	R/W	FFFFh	NMI Watchdog Period  These bits specify the period of the NMI watchdog timer in SYSCLK cycles. Writing a period value that is smaller than the current counter value will immediately force a reset (NMIWDRS). Reset type: SYSRSn

**3.14.17.7 NMISHDFLG Register (Offset = 6h) [reset = 0h]**

NMISHDFLG is shown in [Figure 3-237](#) and described in [Table 3-265](#).

Return to the [Summary Table](#).

These shadow flag bits are set whenever the corresponding bits in the NMIFLG register are set. The shadow flags are only reset by a power-on reset (POR), but any system or external reset will reset the normal flags. The shadow flags allow NMIs to be tracked across resets.

**Figure 3-237. NMISHDFLG Register**

15	14	13	12	11	10	9	8
RESERVED		SWERR	RESERVED	RESERVED	RESERVED	RESERVED	CLBNMI
R-0-0h		R-0h					R-0h
7	6	5	4	3	2	1	0
RESERVED	PIEVECTERR	RESERVED	RESERVED	FLUNCERR	RAMUNCERR	CLOCKFAIL	RESERVED
R-0h				R-0h	R-0h	R-0h	R-0-0h

**Table 3-265. NMISHDFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R-0	0h	Reserved
13	SWERR	R	0h	Shadow SWERR flag Reset type: PORESETn
12	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	CLBNMI	R	0h	Shadow CLBNMI flag Reset type: PORESETn
7	RESERVED	R	0h	Reserved
6	PIEVECTERR	R	0h	Shadow PIEVECTERR flag Reset type: PORESETn
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	FLUNCERR	R	0h	Shadow FLUNCERR flag Reset type: PORESETn
2	RAMUNCERR	R	0h	Shadow RAMUNCERR flag Reset type: PORESETn
1	CLOCKFAIL	R	0h	Shadow CLOCKFAIL flag Reset type: PORESETn
0	RESERVED	R-0	0h	Reserved

### 3.14.18 PERIPH\_AC\_REGS Registers

Table 3-266 lists the PERIPH\_AC\_REGS registers. All register offset addresses not listed in Table 3-266 should be considered as reserved locations and the register contents should not be modified.

**Table 3-266. PERIPH\_AC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	ADCA_AC	ADCA Master Access Control Register	EALLOW	<a href="#">Go</a>
2h	ADCB_AC	ADCB Master Access Control Register	EALLOW	<a href="#">Go</a>
4h	ADCC_AC	ADCC Master Access Control Register	EALLOW	<a href="#">Go</a>
10h	CMPSS1_AC	CMPSS1 Master Access Control Register	EALLOW	<a href="#">Go</a>
12h	CMPSS2_AC	CMPSS2 Master Access Control Register	EALLOW	<a href="#">Go</a>
14h	CMPSS3_AC	CMPSS3 Master Access Control Register	EALLOW	<a href="#">Go</a>
16h	CMPSS4_AC	CMPSS4 Master Access Control Register	EALLOW	<a href="#">Go</a>
18h	CMPSS5_AC	CMPSS5 Master Access Control Register	EALLOW	<a href="#">Go</a>
1Ah	CMPSS6_AC	CMPSS6 Master Access Control Register	EALLOW	<a href="#">Go</a>
1Ch	CMPSS7_AC	CMPSS7 Master Access Control Register	EALLOW	<a href="#">Go</a>
28h	DACA_AC	DACA Master Access Control Register	EALLOW	<a href="#">Go</a>
2Ah	DACB_AC	DACB Master Access Control Register	EALLOW	<a href="#">Go</a>
38h	PGA1_AC	PGAA Master Access Control Register	EALLOW	<a href="#">Go</a>
3Ah	PGA2_AC	PGAB Master Access Control Register	EALLOW	<a href="#">Go</a>
3Ch	PGA3_AC	PGAC Master Access Control Register	EALLOW	<a href="#">Go</a>
3Eh	PGA4_AC	PGAD Master Access Control Register	EALLOW	<a href="#">Go</a>
40h	PGA5_AC	PGAE Master Access Control Register	EALLOW	<a href="#">Go</a>
42h	PGA6_AC	PGAF Master Access Control Register	EALLOW	<a href="#">Go</a>
44h	PGA7_AC	PGAG Master Access Control Register	EALLOW	<a href="#">Go</a>
48h	EPWM1_AC	EPWM1 Master Access Control Register	EALLOW	<a href="#">Go</a>
4Ah	EPWM2_AC	EPWM2 Master Access Control Register	EALLOW	<a href="#">Go</a>
4Ch	EPWM3_AC	EPWM3 Master Access Control Register	EALLOW	<a href="#">Go</a>
4Eh	EPWM4_AC	EPWM4 Master Access Control Register	EALLOW	<a href="#">Go</a>
50h	EPWM5_AC	EPWM5 Master Access Control Register	EALLOW	<a href="#">Go</a>
52h	EPWM6_AC	EPWM6 Master Access Control Register	EALLOW	<a href="#">Go</a>
54h	EPWM7_AC	EPWM7 Master Access Control Register	EALLOW	<a href="#">Go</a>
56h	EPWM8_AC	EPWM8 Master Access Control Register	EALLOW	<a href="#">Go</a>
70h	EQEP1_AC	EQEP1 Master Access Control Register	EALLOW	<a href="#">Go</a>
72h	EQEP2_AC	EQEP2 Master Access Control Register	EALLOW	<a href="#">Go</a>
80h	ECAP1_AC	ECAP1 Master Access Control Register	EALLOW	<a href="#">Go</a>
82h	ECAP2_AC	ECAP2 Master Access Control Register	EALLOW	<a href="#">Go</a>
84h	ECAP3_AC	ECAP3 Master Access Control Register	EALLOW	<a href="#">Go</a>
86h	ECAP4_AC	ECAP4 Master Access Control Register	EALLOW	<a href="#">Go</a>
88h	ECAP5_AC	ECAP5 Master Access Control Register	EALLOW	<a href="#">Go</a>
8Ah	ECAP6_AC	ECAP6 Master Access Control Register	EALLOW	<a href="#">Go</a>
8Ch	ECAP7_AC	ECAP7 Master Access Control Register	EALLOW	<a href="#">Go</a>
A8h	SDFM1_AC	SDFM1 Master Access Control Register	EALLOW	<a href="#">Go</a>
B0h	CLB1_AC	CLB1 Master Access Control Register	EALLOW	<a href="#">Go</a>
B2h	CLB2_AC	CLB2 Master Access Control Register	EALLOW	<a href="#">Go</a>
B4h	CLB3_AC	CLB3 Master Access Control Register	EALLOW	<a href="#">Go</a>
B6h	CLB4_AC	CLB4 Master Access Control Register	EALLOW	<a href="#">Go</a>
C0h	CLA1PROMCRC_AC	CLA1PROMCRC Master Access Control Register	EALLOW	<a href="#">Go</a>
110h	SPIA_AC	SPIA Master Access Control Register	EALLOW	<a href="#">Go</a>



**Table 3-266. PERIPH\_AC\_REGS Registers (continued)**

Offset	Acronym	Register Name	Write Protection	Section
112h	SPIB_AC	SPIB Master Access Control Register	EALLOW	<a href="#">Go</a>
130h	PMBUS_A_AC	PMBUSA Master Access Control Register	EALLOW	<a href="#">Go</a>
138h	LIN_A_AC	LINA Master Access Control Register	EALLOW	<a href="#">Go</a>
140h	DCANA_AC	DCANA Master Access Control Register	EALLOW	<a href="#">Go</a>
142h	DCANB_AC	DCANB Master Access Control Register	EALLOW	<a href="#">Go</a>
158h	FSIATX_AC	FSIA Master Access Control Register	EALLOW	<a href="#">Go</a>
15Ah	FSIARX_AC	FSIB Master Access Control Register	EALLOW	<a href="#">Go</a>
1AAh	HRPWM_A_AC	HRPWM Master Access Control Register	EALLOW	<a href="#">Go</a>
1FEh	PERIPH_AC_LOCK	Lock Register to stop Write access to peripheral Access register.	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 3-267](#) shows the codes that are used for access types in this section.

**Table 3-267. PERIPH\_AC\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
WOnce	WOnce	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.18.1 ADCA\_AC Register (Offset = 0h) [reset = 3Fh]**

ADCA\_AC is shown in [Figure 3-238](#) and described in [Table 3-268](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-238. ADCA\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-268. ADCA\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.2 ADCB\_AC Register (Offset = 2h) [reset = 3Fh]**

ADCB\_AC is shown in [Figure 3-239](#) and described in [Table 3-269](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-239. ADCB\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-269. ADCB\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.3 ADCC\_AC Register (Offset = 4h) [reset = 3Fh]**

ADCC\_AC is shown in [Figure 3-240](#) and described in [Table 3-270](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-240. ADCC\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-270. ADCC\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

### 3.14.18.4 CMPSS1\_AC Register (Offset = 10h) [reset = 3Fh]

CMPSS1\_AC is shown in [Figure 3-241](#) and described in [Table 3-271](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-241. CMPSS1\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-271. CMPSS1\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

### 3.14.18.5 CMPSS2\_AC Register (Offset = 12h) [reset = 3Fh]

CMPSS2\_AC is shown in [Figure 3-242](#) and described in [Table 3-272](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-242. CMPSS2\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-272. CMPSS2\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.6 CMPSS3\_AC Register (Offset = 14h) [reset = 3Fh]**

CMPSS3\_AC is shown in [Figure 3-243](#) and described in [Table 3-273](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-243. CMPSS3\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-273. CMPSS3\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.7 CMPSS4\_AC Register (Offset = 16h) [reset = 3Fh]**

CMPSS4\_AC is shown in [Figure 3-244](#) and described in [Table 3-274](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-244. CMPSS4\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-274. CMPSS4\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn



**3.14.18.8 CMPSS5\_AC Register (Offset = 18h) [reset = 3Fh]**

CMPSS5\_AC is shown in [Figure 3-245](#) and described in [Table 3-275](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-245. CMPSS5\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-275. CMPSS5\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.9 CMPSS6\_AC Register (Offset = 1Ah) [reset = 3Fh]**

CMPSS6\_AC is shown in [Figure 3-246](#) and described in [Table 3-276](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-246. CMPSS6\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-276. CMPSS6\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.10 CMPSS7\_AC Register (Offset = 1Ch) [reset = 3Fh]**

CMPSS7\_AC is shown in [Figure 3-247](#) and described in [Table 3-277](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-247. CMPSS7\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-277. CMPSS7\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.11 DACA\_AC Register (Offset = 28h) [reset = 3Fh]**

DACA\_AC is shown in [Figure 3-248](#) and described in [Table 3-278](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-248. DACA\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-278. DACA\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.12 DACB\_AC Register (Offset = 2Ah) [reset = 3Fh]**

DACB\_AC is shown in [Figure 3-249](#) and described in [Table 3-279](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-249. DACB\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-279. DACB\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.13 PGA1\_AC Register (Offset = 38h) [reset = 3Fh]**

PGA1\_AC is shown in [Figure 3-250](#) and described in [Table 3-280](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-250. PGA1\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-280. PGA1\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.14 PGA2\_AC Register (Offset = 3Ah) [reset = 3Fh]**

PGA2\_AC is shown in [Figure 3-251](#) and described in [Table 3-281](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-251. PGA2\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-281. PGA2\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.15 PGA3\_AC Register (Offset = 3Ch) [reset = 3Fh]**

PGA3\_AC is shown in [Figure 3-252](#) and described in [Table 3-282](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-252. PGA3\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-282. PGA3\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn



**3.14.18.16 PGA4\_AC Register (Offset = 3Eh) [reset = 3Fh]**

PGA4\_AC is shown in [Figure 3-253](#) and described in [Table 3-283](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-253. PGA4\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-283. PGA4\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.17 PGA5\_AC Register (Offset = 40h) [reset = 3Fh]**

PGA5\_AC is shown in [Figure 3-254](#) and described in [Table 3-284](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-254. PGA5\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-284. PGA5\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.18 PGA6\_AC Register (Offset = 42h) [reset = 3Fh]**

PGA6\_AC is shown in [Figure 3-255](#) and described in [Table 3-285](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-255. PGA6\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-285. PGA6\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.19 PGA7\_AC Register (Offset = 44h) [reset = 3Fh]**

PGA7\_AC is shown in [Figure 3-256](#) and described in [Table 3-286](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-256. PGA7\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-286. PGA7\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.20 EPWM1\_AC Register (Offset = 48h) [reset = 3Fh]**

EPWM1\_AC is shown in [Figure 3-257](#) and described in [Table 3-287](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-257. EPWM1\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-287. EPWM1\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.21 EPWM2\_AC Register (Offset = 4Ah) [reset = 3Fh]**

EPWM2\_AC is shown in [Figure 3-258](#) and described in [Table 3-288](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-258. EPWM2\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-288. EPWM2\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.22 EPWM3\_AC Register (Offset = 4Ch) [reset = 3Fh]**

EPWM3\_AC is shown in [Figure 3-259](#) and described in [Table 3-289](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-259. EPWM3\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-289. EPWM3\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.23 EPWM4\_AC Register (Offset = 4Eh) [reset = 3Fh]**

EPWM4\_AC is shown in [Figure 3-260](#) and described in [Table 3-290](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-260. EPWM4\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-290. EPWM4\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn



**3.14.18.24 EPWM5\_AC Register (Offset = 50h) [reset = 3Fh]**

EPWM5\_AC is shown in [Figure 3-261](#) and described in [Table 3-291](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-261. EPWM5\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-291. EPWM5\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.25 EPWM6\_AC Register (Offset = 52h) [reset = 3Fh]**

EPWM6\_AC is shown in [Figure 3-262](#) and described in [Table 3-292](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-262. EPWM6\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-292. EPWM6\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.26 EPWM7\_AC Register (Offset = 54h) [reset = 3Fh]**

EPWM7\_AC is shown in [Figure 3-263](#) and described in [Table 3-293](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-263. EPWM7\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-293. EPWM7\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.27 EPWM8\_AC Register (Offset = 56h) [reset = 3Fh]**

EPWM8\_AC is shown in [Figure 3-264](#) and described in [Table 3-294](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-264. EPWM8\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-294. EPWM8\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.28 EQEP1\_AC Register (Offset = 70h) [reset = 3Fh]**

EQEP1\_AC is shown in [Figure 3-265](#) and described in [Table 3-295](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-265. EQEP1\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-295. EQEP1\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.29 EQEP2\_AC Register (Offset = 72h) [reset = 3Fh]**

EQEP2\_AC is shown in [Figure 3-266](#) and described in [Table 3-296](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-266. EQEP2\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-296. EQEP2\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.30 ECAP1\_AC Register (Offset = 80h) [reset = 3Fh]**

ECAP1\_AC is shown in [Figure 3-267](#) and described in [Table 3-297](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-267. ECAP1\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-297. ECAP1\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.31 ECAP2\_AC Register (Offset = 82h) [reset = 3Fh]**

ECAP2\_AC is shown in [Figure 3-268](#) and described in [Table 3-298](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-268. ECAP2\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-298. ECAP2\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn



**3.14.18.32 ECAP3\_AC Register (Offset = 84h) [reset = 3Fh]**

ECAP3\_AC is shown in [Figure 3-269](#) and described in [Table 3-299](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-269. ECAP3\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-299. ECAP3\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.33 ECAP4\_AC Register (Offset = 86h) [reset = 3Fh]**

ECAP4\_AC is shown in [Figure 3-270](#) and described in [Table 3-300](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-270. ECAP4\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-300. ECAP4\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.34 ECAP5\_AC Register (Offset = 88h) [reset = 3Fh]**

ECAP5\_AC is shown in [Figure 3-271](#) and described in [Table 3-301](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-271. ECAP5\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-301. ECAP5\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.35 ECAP6\_AC Register (Offset = 8Ah) [reset = 3Fh]**

ECAP6\_AC is shown in [Figure 3-272](#) and described in [Table 3-302](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-272. ECAP6\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-302. ECAP6\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

### 3.14.18.36 ECAP7\_AC Register (Offset = 8Ch) [reset = 3Fh]

ECAP7\_AC is shown in [Figure 3-273](#) and described in [Table 3-303](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-273. ECAP7\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-303. ECAP7\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.37 SDFM1\_AC Register (Offset = A8h) [reset = 3Fh]**

SDFM1\_AC is shown in [Figure 3-274](#) and described in [Table 3-304](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-274. SDFM1\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-304. SDFM1\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.38 CLB1\_AC Register (Offset = B0h) [reset = 3Fh]**

CLB1\_AC is shown in [Figure 3-275](#) and described in [Table 3-305](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-275. CLB1\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		CLA1_ACC		CPU1_ACC	
R-0-0h				R/W-3h		R/W-3h	

**Table 3-305. CLB1\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	RESERVED	R/W	3h	Reserved
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.39 CLB2\_AC Register (Offset = B2h) [reset = 3Fh]**

CLB2\_AC is shown in [Figure 3-276](#) and described in [Table 3-306](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-276. CLB2\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		CLA1_ACC		CPU1_ACC	
R-0-0h				R/W-3h		R/W-3h	

**Table 3-306. CLB2\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	RESERVED	R/W	3h	Reserved
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn



**3.14.18.40 CLB3\_AC Register (Offset = B4h) [reset = 3Fh]**

CLB3\_AC is shown in [Figure 3-277](#) and described in [Table 3-307](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-277. CLB3\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		CLA1_ACC		CPU1_ACC	
R-0-0h				R/W-3h		R/W-3h	

**Table 3-307. CLB3\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	RESERVED	R/W	3h	Reserved
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.41 CLB4\_AC Register (Offset = B6h) [reset = 3Fh]**

CLB4\_AC is shown in [Figure 3-278](#) and described in [Table 3-308](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-278. CLB4\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		CLA1_ACC		CPU1_ACC	
R-0-0h				R/W-3h		R/W-3h	

**Table 3-308. CLB4\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	RESERVED	R/W	3h	Reserved
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.42 CLA1PROMCRC\_AC Register (Offset = C0h) [reset = 3Fh]**

CLA1PROMCRC\_AC is shown in [Figure 3-279](#) and described in [Table 3-309](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-279. CLA1PROMCRC\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		RESERVED		CLA1_ACC		CPU1_ACC	
R-0-0h				R/W-3h		R/W-3h	

**Table 3-309. CLA1PROMCRC\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	RESERVED	R/W	3h	Reserved
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.43 SPIA\_AC Register (Offset = 110h) [reset = 3Fh]**

SPIA\_AC is shown in [Figure 3-280](#) and described in [Table 3-310](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-280. SPIA\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-310. SPIA\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

### 3.14.18.44 SPIB\_AC Register (Offset = 112h) [reset = 3Fh]

SPIB\_AC is shown in [Figure 3-281](#) and described in [Table 3-311](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-281. SPIB\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-311. SPIB\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.45 PMBUS\_A\_AC Register (Offset = 130h) [reset = 3Fh]**

PMBUS\_A\_AC is shown in [Figure 3-282](#) and described in [Table 3-312](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-282. PMBUS\_A\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-312. PMBUS\_A\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.46 LIN\_A\_AC Register (Offset = 138h) [reset = 3Fh]**

LIN\_A\_AC is shown in [Figure 3-283](#) and described in [Table 3-313](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-283. LIN\_A\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-313. LIN\_A\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.47 DCANA\_AC Register (Offset = 140h) [reset = 3Fh]**

DCANA\_AC is shown in [Figure 3-284](#) and described in [Table 3-314](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-284. DCANA\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		RESERVED		CPU1_ACC	
R-0-0h		R/W-3h				R/W-3h	

**Table 3-314. DCANA\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	RESERVED	R/W	3h	Reserved
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn



**3.14.18.48 DCANB\_AC Register (Offset = 142h) [reset = 3Fh]**

DCANB\_AC is shown in [Figure 3-285](#) and described in [Table 3-315](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-285. DCANB\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		RESERVED		CPU1_ACC	
R-0-0h		R/W-3h				R/W-3h	

**Table 3-315. DCANB\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	RESERVED	R/W	3h	Reserved
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.49 FSIATX\_AC Register (Offset = 158h) [reset = 3Fh]**

FSIATX\_AC is shown in [Figure 3-286](#) and described in [Table 3-316](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-286. FSIATX\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-316. FSIATX\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.50 FSIARX\_AC Register (Offset = 15Ah) [reset = 3Fh]**

FSIARX\_AC is shown in [Figure 3-287](#) and described in [Table 3-317](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-287. FSIARX\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-317. FSIARX\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.51 HRPWM\_A\_AC Register (Offset = 1AAh) [reset = 3Fh]**

HRPWM\_A\_AC is shown in [Figure 3-288](#) and described in [Table 3-318](#).

Return to the [Summary Table](#).

Based on control settings allows Full, Protected Read, No Access to peripheral from corresponding connected master.

**Figure 3-288. HRPWM\_A\_AC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		DMA1_ACC		CLA1_ACC		CPU1_ACC	
R-0-0h		R/W-3h		R/W-3h		R/W-3h	

**Table 3-318. HRPWM\_A\_AC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-4	DMA1_ACC	R/W	3h	Defines Access control definition for the DMA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
3-2	CLA1_ACC	R/W	3h	Defines Access control definition for the CLA1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn
1-0	CPU1_ACC	R/W	3h	Defines Access control definition for the CPU1 as: 11: Full Access for both read and Write 10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access 01: Reserved 00: No Read/Write Access to peripheral Reset type: XRSn

**3.14.18.52 PERIPH\_AC\_LOCK Register (Offset = 1FEh) [reset = 0h]**

PERIPH\_AC\_LOCK is shown in [Figure 3-289](#) and described in [Table 3-319](#).

Return to the [Summary Table](#).

Based on status bit control the Access registers are either RD/WR or RD only.

**Figure 3-289. PERIPH\_AC\_LOCK Register**

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

**Table 3-319. PERIPH\_AC\_LOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	LOCK_AC_WR	R/WOnce	0h	Defines Access control definition for the CPU1 as: 1: Access Control registers are Read Only 0: Read/Write Access allowed to Access Control registers. Writing '1' sets the bit, writing '0' has no effect. Reset type: SYSRSn

### 3.14.19 PIE\_CTRL\_REGS Registers

Table 3-320 lists the PIE\_CTRL\_REGS registers. All register offset addresses not listed in Table 3-320 should be considered as reserved locations and the register contents should not be modified.

**Table 3-320. PIE\_CTRL\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	PIECTRL	ePIE Control Register		<a href="#">Go</a>
1h	PIEACK	Interrupt Acknowledge Register		<a href="#">Go</a>
2h	PIEIER1	Interrupt Group 1 Enable Register		<a href="#">Go</a>
3h	PIEIFR1	Interrupt Group 1 Flag Register		<a href="#">Go</a>
4h	PIEIER2	Interrupt Group 2 Enable Register		<a href="#">Go</a>
5h	PIEIFR2	Interrupt Group 2 Flag Register		<a href="#">Go</a>
6h	PIEIER3	Interrupt Group 3 Enable Register		<a href="#">Go</a>
7h	PIEIFR3	Interrupt Group 3 Flag Register		<a href="#">Go</a>
8h	PIEIER4	Interrupt Group 4 Enable Register		<a href="#">Go</a>
9h	PIEIFR4	Interrupt Group 4 Flag Register		<a href="#">Go</a>
Ah	PIEIER5	Interrupt Group 5 Enable Register		<a href="#">Go</a>
Bh	PIEIFR5	Interrupt Group 5 Flag Register		<a href="#">Go</a>
Ch	PIEIER6	Interrupt Group 6 Enable Register		<a href="#">Go</a>
Dh	PIEIFR6	Interrupt Group 6 Flag Register		<a href="#">Go</a>
Eh	PIEIER7	Interrupt Group 7 Enable Register		<a href="#">Go</a>
Fh	PIEIFR7	Interrupt Group 7 Flag Register		<a href="#">Go</a>
10h	PIEIER8	Interrupt Group 8 Enable Register		<a href="#">Go</a>
11h	PIEIFR8	Interrupt Group 8 Flag Register		<a href="#">Go</a>
12h	PIEIER9	Interrupt Group 9 Enable Register		<a href="#">Go</a>
13h	PIEIFR9	Interrupt Group 9 Flag Register		<a href="#">Go</a>
14h	PIEIER10	Interrupt Group 10 Enable Register		<a href="#">Go</a>
15h	PIEIFR10	Interrupt Group 10 Flag Register		<a href="#">Go</a>
16h	PIEIER11	Interrupt Group 11 Enable Register		<a href="#">Go</a>
17h	PIEIFR11	Interrupt Group 11 Flag Register		<a href="#">Go</a>
18h	PIEIER12	Interrupt Group 12 Enable Register		<a href="#">Go</a>
19h	PIEIFR12	Interrupt Group 12 Flag Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-321 shows the codes that are used for access types in this section.

**Table 3-321. PIE\_CTRL\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		

**Table 3-321. PIE\_CTRL\_REGS Access Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.19.1 PIECTRL Register (Offset = 0h) [reset = 0h]**

PIECTRL is shown in [Figure 3-290](#) and described in [Table 3-322](#).

Return to the [Summary Table](#).

ePIE Control Register

**Figure 3-290. PIECTRL Register**

15	14	13	12	11	10	9	8
PIEVECT							
R-0h							
7	6	5	4	3	2	1	0
PIEVECT							ENPIE
R-0h							R/W-0h

**Table 3-322. PIECTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	PIEVECT	R	0h	These bits indicate the vector address of the vector fetched from the ePIE vector table. The least significant bit of the address is ignored and only bits 1 to 15 of the address are shown. The vector value can be read by the user to determine which interrupt generated the vector fetch.  Note: When a NMI is serviced, the PIEVECT bit-field does not reflect the vector as it does for other interrupts. Reset type: SYSRSn
0	ENPIE	R/W	0h	Enable vector fetching from ePIE block. This bit must be set to 1 for peripheral interrupts to work. All ePIE registers (PIEACK, PIEIFR, PIEIER) can be accessed even when the ePIE block is disabled. Reset type: SYSRSn



### 3.14.19.2 PIEACK Register (Offset = 1h) [reset = 0h]

PIEACK is shown in [Figure 3-291](#) and described in [Table 3-323](#).

Return to the [Summary Table](#).

#### Acknowledge Register

When an interrupt propagates from the ePIE to a CPU interrupt line, the interrupt group's PIEACK bit is set. This prevents other interrupts in that group from propagating to the CPU while the first interrupt is handled. Writing a 1 to a PIEACK bit clears it and allows another interrupt from the corresponding group to propagate. ISRs for PIE interrupts should clear the group's PIEACK bit before returning from the interrupt.

Writes of 0 are ignored.

**Figure 3-291. PIEACK Register**

15	14	13	12	11	10	9	8
RESERVED				ACK12	ACK11	ACK10	ACK9
R-0-0h				R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h
7	6	5	4	3	2	1	0
ACK8	ACK7	ACK6	ACK5	ACK4	ACK3	ACK2	ACK1
R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h

**Table 3-323. PIEACK Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11	ACK12	R/W1S	0h	Acknowledge PIE Interrupt Group 12 Reset type: SYSRSn
10	ACK11	R/W1S	0h	Acknowledge PIE Interrupt Group 11 Reset type: SYSRSn
9	ACK10	R/W1S	0h	Acknowledge PIE Interrupt Group 10 Reset type: SYSRSn
8	ACK9	R/W1S	0h	Acknowledge PIE Interrupt Group 9 Reset type: SYSRSn
7	ACK8	R/W1S	0h	Acknowledge PIE Interrupt Group 8 Reset type: SYSRSn
6	ACK7	R/W1S	0h	Acknowledge PIE Interrupt Group 7 Reset type: SYSRSn
5	ACK6	R/W1S	0h	Acknowledge PIE Interrupt Group 6 Reset type: SYSRSn
4	ACK5	R/W1S	0h	Acknowledge PIE Interrupt Group 5 Reset type: SYSRSn
3	ACK4	R/W1S	0h	Acknowledge PIE Interrupt Group 4 Reset type: SYSRSn
2	ACK3	R/W1S	0h	Acknowledge PIE Interrupt Group 3 Reset type: SYSRSn
1	ACK2	R/W1S	0h	Acknowledge PIE Interrupt Group 2 Reset type: SYSRSn
0	ACK1	R/W1S	0h	Acknowledge PIE Interrupt Group 1 Reset type: SYSRSn

### 3.14.19.3 PIEIER1 Register (Offset = 2h) [reset = 0h]

PIEIER1 is shown in [Figure 3-292](#) and described in [Table 3-324](#).

Return to the [Summary Table](#).

Interrupt Group 1 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-292. PIEIER1 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-324. PIEIER1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 1.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 1.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 1.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 1.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 1.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 1.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 1.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 1.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 1.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 1.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 1.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 1.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 1.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 1.3 Reset type: SYSRSn

**Table 3-324. PIEIER1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 1.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 1.1 Reset type: SYSRSn

### 3.14.19.4 PIEIFR1 Register (Offset = 3h) [reset = 0h]

PIEIFR1 is shown in [Figure 3-293](#) and described in [Table 3-325](#).

Return to the [Summary Table](#).

#### Interrupt Group 1 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-293. PIEIFR1 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-325. PIEIFR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 1.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 1.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 1.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 1.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 1.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 1.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 1.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 1.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 1.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 1.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 1.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 1.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 1.4 Reset type: SYSRSn

**Table 3-325. PIEIFR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 1.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 1.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 1.1 Reset type: SYSRSn

### 3.14.19.5 PIEIER2 Register (Offset = 4h) [reset = 0h]

PIEIER2 is shown in [Figure 3-294](#) and described in [Table 3-326](#).

Return to the [Summary Table](#).

Interrupt Group 2 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-294. PIEIER2 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-326. PIEIER2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 2.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 2.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 2.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 2.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 2.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 2.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 2.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 2.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 2.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 2.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 2.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 2.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 2.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 2.3 Reset type: SYSRSn

**Table 3-326. PIEIER2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 2.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 2.1 Reset type: SYSRSn

### 3.14.19.6 PIEIFR2 Register (Offset = 5h) [reset = 0h]

PIEIFR2 is shown in [Figure 3-295](#) and described in [Table 3-327](#).

Return to the [Summary Table](#).

#### Interrupt Group 2 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-295. PIEIFR2 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-327. PIEIFR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 2.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 2.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 2.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 2.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 2.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 2.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 2.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 2.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 2.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 2.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 2.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 2.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 2.4 Reset type: SYSRSn



**Table 3-327. PIEIFR2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 2.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 2.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 2.1 Reset type: SYSRSn

### 3.14.19.7 PIEIER3 Register (Offset = 6h) [reset = 0h]

PIEIER3 is shown in [Figure 3-296](#) and described in [Table 3-328](#).

Return to the [Summary Table](#).

Interrupt Group 3 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-296. PIEIER3 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-328. PIEIER3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 3.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 3.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 3.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 3.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 3.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 3.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 3.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 3.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 3.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 3.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 3.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 3.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 3.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 3.3 Reset type: SYSRSn

**Table 3-328. PIEIER3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 3.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 3.1 Reset type: SYSRSn

### 3.14.19.8 PIEIFR3 Register (Offset = 7h) [reset = 0h]

PIEIFR3 is shown in [Figure 3-297](#) and described in [Table 3-329](#).

Return to the [Summary Table](#).

#### Interrupt Group 3 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-297. PIEIFR3 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-329. PIEIFR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 3.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 3.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 3.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 3.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 3.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 3.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 3.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 3.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 3.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 3.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 3.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 3.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 3.4 Reset type: SYSRSn

**Table 3-329. PIEIFR3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 3.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 3.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 3.1 Reset type: SYSRSn

### 3.14.19.9 PIEIER4 Register (Offset = 8h) [reset = 0h]

PIEIER4 is shown in [Figure 3-298](#) and described in [Table 3-330](#).

Return to the [Summary Table](#).

Interrupt Group 4 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-298. PIEIER4 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-330. PIEIER4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 4.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 4.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 4.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 4.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 4.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 4.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 4.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 4.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 4.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 4.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 4.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 4.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 4.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 4.3 Reset type: SYSRSn

**Table 3-330. PIEIER4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 4.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 4.1 Reset type: SYSRSn

### 3.14.19.10 PIEIFR4 Register (Offset = 9h) [reset = 0h]

PIEIFR4 is shown in [Figure 3-299](#) and described in [Table 3-331](#).

Return to the [Summary Table](#).

#### Interrupt Group 4 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-299. PIEIFR4 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-331. PIEIFR4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 4.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 4.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 4.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 4.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 4.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 4.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 4.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 4.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 4.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 4.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 4.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 4.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 4.4 Reset type: SYSRSn



**Table 3-331. PIEIFR4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 4.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 4.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 4.1 Reset type: SYSRSn

### 3.14.19.11 PIEIER5 Register (Offset = Ah) [reset = 0h]

PIEIER5 is shown in [Figure 3-300](#) and described in [Table 3-332](#).

Return to the [Summary Table](#).

Interrupt Group 5 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-300. PIEIER5 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-332. PIEIER5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 5.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 5.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 5.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 5.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 5.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 5.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 5.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 5.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 5.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 5.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 5.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 5.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 5.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 5.3 Reset type: SYSRSn

**Table 3-332. PIEIER5 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 5.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 5.1 Reset type: SYSRSn

### 3.14.19.12 PIEIFR5 Register (Offset = Bh) [reset = 0h]

PIEIFR5 is shown in [Figure 3-301](#) and described in [Table 3-333](#).

Return to the [Summary Table](#).

#### Interrupt Group 5 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-301. PIEIFR5 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-333. PIEIFR5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 5.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 5.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 5.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 5.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 5.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 5.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 5.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 5.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 5.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 5.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 5.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 5.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 5.4 Reset type: SYSRSn

**Table 3-333. PIEIFR5 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 5.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 5.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 5.1 Reset type: SYSRSn

### 3.14.19.13 PIEIER6 Register (Offset = Ch) [reset = 0h]

PIEIER6 is shown in [Figure 3-302](#) and described in [Table 3-334](#).

Return to the [Summary Table](#).

Interrupt Group 6 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-302. PIEIER6 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-334. PIEIER6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 6.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 6.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 6.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 6.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 6.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 6.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 6.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 6.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 6.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 6.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 6.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 6.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 6.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 6.3 Reset type: SYSRSn

**Table 3-334. PIEIER6 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 6.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 6.1 Reset type: SYSRSn

### 3.14.19.14 PIEIFR6 Register (Offset = Dh) [reset = 0h]

PIEIFR6 is shown in [Figure 3-303](#) and described in [Table 3-335](#).

Return to the [Summary Table](#).

#### Interrupt Group 6 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-303. PIEIFR6 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-335. PIEIFR6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 6.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 6.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 6.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 6.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 6.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 6.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 6.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 6.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 6.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 6.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 6.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 6.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 6.4 Reset type: SYSRSn



**Table 3-335. PIEIFR6 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 6.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 6.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 6.1 Reset type: SYSRSn

### 3.14.19.15 PIEIER7 Register (Offset = Eh) [reset = 0h]

PIEIER7 is shown in [Figure 3-304](#) and described in [Table 3-336](#).

Return to the [Summary Table](#).

Interrupt Group 7 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-304. PIEIER7 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-336. PIEIER7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 7.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 7.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 7.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 7.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 7.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 7.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 7.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 7.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 7.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 7.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 7.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 7.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 7.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 7.3 Reset type: SYSRSn

**Table 3-336. PIEIER7 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 7.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 7.1 Reset type: SYSRSn

### 3.14.19.16 PIEIFR7 Register (Offset = Fh) [reset = 0h]

PIEIFR7 is shown in [Figure 3-305](#) and described in [Table 3-337](#).

Return to the [Summary Table](#).

#### Interrupt Group 7 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-305. PIEIFR7 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-337. PIEIFR7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 7.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 7.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 7.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 7.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 7.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 7.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 7.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 7.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 7.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 7.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 7.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 7.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 7.4 Reset type: SYSRSn

**Table 3-337. PIEIFR7 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 7.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 7.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 7.1 Reset type: SYSRSn

### 3.14.19.17 PIEIER8 Register (Offset = 10h) [reset = 0h]

PIEIER8 is shown in [Figure 3-306](#) and described in [Table 3-338](#).

Return to the [Summary Table](#).

Interrupt Group 8 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-306. PIEIER8 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-338. PIEIER8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 8.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 8.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 8.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 8.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 8.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 8.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 8.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 8.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 8.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 8.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 8.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 8.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 8.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 8.3 Reset type: SYSRSn

**Table 3-338. PIEIER8 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 8.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 8.1 Reset type: SYSRSn

### 3.14.19.18 PIEIFR8 Register (Offset = 11h) [reset = 0h]

PIEIFR8 is shown in [Figure 3-307](#) and described in [Table 3-339](#).

Return to the [Summary Table](#).

#### Interrupt Group 8 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-307. PIEIFR8 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-339. PIEIFR8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 8.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 8.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 8.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 8.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 8.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 8.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 8.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 8.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 8.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 8.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 8.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 8.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 8.4 Reset type: SYSRSn



**Table 3-339. PIEIFR8 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 8.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 8.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 8.1 Reset type: SYSRSn

### 3.14.19.19 PIEIER9 Register (Offset = 12h) [reset = 0h]

PIEIER9 is shown in [Figure 3-308](#) and described in [Table 3-340](#).

Return to the [Summary Table](#).

Interrupt Group 9 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-308. PIEIER9 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-340. PIEIER9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 9.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 9.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 9.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 9.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 9.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 9.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 9.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 9.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 9.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 9.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 9.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 9.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 9.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 9.3 Reset type: SYSRSn

**Table 3-340. PIEIER9 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 9.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 9.1 Reset type: SYSRSn

### 3.14.19.20 PIEIFR9 Register (Offset = 13h) [reset = 0h]

PIEIFR9 is shown in [Figure 3-309](#) and described in [Table 3-341](#).

Return to the [Summary Table](#).

#### Interrupt Group 9 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-309. PIEIFR9 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-341. PIEIFR9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 9.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 9.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 9.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 9.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 9.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 9.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 9.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 9.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 9.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 9.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 9.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 9.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 9.4 Reset type: SYSRSn

**Table 3-341. PIEIFR9 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 9.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 9.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 9.1 Reset type: SYSRSn

### 3.14.19.21 PIEIER10 Register (Offset = 14h) [reset = 0h]

PIEIER10 is shown in [Figure 3-310](#) and described in [Table 3-342](#).

Return to the [Summary Table](#).

Interrupt Group 10 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-310. PIEIER10 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-342. PIEIER10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 10.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 10.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 10.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 10.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 10.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 10.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 10.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 10.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 10.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 10.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 10.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 10.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 10.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 10.3 Reset type: SYSRSn

**Table 3-342. PIEIER10 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 10.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 10.1 Reset type: SYSRSn

### 3.14.19.22 PIEIFR10 Register (Offset = 15h) [reset = 0h]

PIEIFR10 is shown in [Figure 3-311](#) and described in [Table 3-343](#).

Return to the [Summary Table](#).

#### Interrupt Group 10 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-311. PIEIFR10 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-343. PIEIFR10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 10.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 10.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 10.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 10.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 10.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 10.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 10.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 10.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 10.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 10.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 10.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 10.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 10.4 Reset type: SYSRSn



**Table 3-343. PIEIFR10 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 10.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 10.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 10.1 Reset type: SYSRSn

### 3.14.19.23 PIEIER11 Register (Offset = 16h) [reset = 0h]

PIEIER11 is shown in [Figure 3-312](#) and described in [Table 3-344](#).

Return to the [Summary Table](#).

Interrupt Group 11 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-312. PIEIER11 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-344. PIEIER11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 11.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 11.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 11.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 11.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 11.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 11.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 11.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 11.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 11.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 11.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 11.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 11.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 11.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 11.3 Reset type: SYSRSn

**Table 3-344. PIEIER11 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 11.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 11.1 Reset type: SYSRSn

### 3.14.19.24 PIEIFR11 Register (Offset = 17h) [reset = 0h]

PIEIFR11 is shown in [Figure 3-313](#) and described in [Table 3-345](#).

Return to the [Summary Table](#).

#### Interrupt Group 11 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-313. PIEIFR11 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-345. PIEIFR11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 11.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 11.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 11.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 11.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 11.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 11.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 11.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 11.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 11.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 11.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 11.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 11.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 11.4 Reset type: SYSRSn

**Table 3-345. PIEIFR11 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 11.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 11.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 11.1 Reset type: SYSRSn

### 3.14.19.25 PIEIER12 Register (Offset = 18h) [reset = 0h]

PIEIER12 is shown in [Figure 3-314](#) and described in [Table 3-346](#).

Return to the [Summary Table](#).

Interrupt Group 12 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-314. PIEIER12 Register**

15		14		13		12		11		10		9		8	
INTx16		INTx15		INTx14		INTx13		INTx12		INTx11		INTx10		INTx9	
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	
INTx8		INTx7		INTx6		INTx5		INTx4		INTx3		INTx2		INTx1	
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 3-346. PIEIER12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Enable for Interrupt 12.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Enable for Interrupt 12.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Enable for Interrupt 12.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Enable for Interrupt 12.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Enable for Interrupt 12.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Enable for Interrupt 12.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Enable for Interrupt 12.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Enable for Interrupt 12.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Enable for Interrupt 12.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 12.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 12.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 12.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 12.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 12.3 Reset type: SYSRSn

**Table 3-346. PIEIER12 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INTx2	R/W	0h	Enable for Interrupt 12.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 12.1 Reset type: SYSRSn

### 3.14.19.26 PIEIFR12 Register (Offset = 19h) [reset = 0h]

PIEIFR12 is shown in [Figure 3-315](#) and described in [Table 3-347](#).

Return to the [Summary Table](#).

#### Interrupt Group 12 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-315. PIEIFR12 Register**

15	14	13	12	11	10	9	8
INTx16	INTx15	INTx14	INTx13	INTx12	INTx11	INTx10	INTx9
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
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
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 3-347. PIEIFR12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Flag for Interrupt 12.16 Reset type: SYSRSn
14	INTx15	R/W	0h	Flag for Interrupt 12.15 Reset type: SYSRSn
13	INTx14	R/W	0h	Flag for Interrupt 12.14 Reset type: SYSRSn
12	INTx13	R/W	0h	Flag for Interrupt 12.13 Reset type: SYSRSn
11	INTx12	R/W	0h	Flag for Interrupt 12.12 Reset type: SYSRSn
10	INTx11	R/W	0h	Flag for Interrupt 12.11 Reset type: SYSRSn
9	INTx10	R/W	0h	Flag for Interrupt 12.10 Reset type: SYSRSn
8	INTx9	R/W	0h	Flag for Interrupt 12.9 Reset type: SYSRSn
7	INTx8	R/W	0h	Flag for Interrupt 12.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 12.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 12.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 12.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 12.4 Reset type: SYSRSn



**Table 3-347. PIEIFR12 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTx3	R/W	0h	Flag for Interrupt 12.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 12.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 12.1 Reset type: SYSRSn

### 3.14.20 ROM\_PREFETCH\_REGS Registers

Table 3-348 lists the ROM\_PREFETCH\_REGS registers. All register offset addresses not listed in Table 3-348 should be considered as reserved locations and the register contents should not be modified.

**Table 3-348. ROM\_PREFETCH\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
--------	---------	---------------	------------------	---------

Complex bit access types are encoded to fit into small table cells. Table 3-349 shows the codes that are used for access types in this section.

**Table 3-349. ROM\_PREFETCH\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.21 ROM\_WAIT\_STATE\_REGS Registers

Table 3-350 lists the ROM\_WAIT\_STATE\_REGS registers. All register offset addresses not listed in Table 3-350 should be considered as reserved locations and the register contents should not be modified.

**Table 3-350. ROM\_WAIT\_STATE\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
--------	---------	---------------	------------------	---------

Complex bit access types are encoded to fit into small table cells. Table 3-351 shows the codes that are used for access types in this section.

**Table 3-351. ROM\_WAIT\_STATE\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.14.22 UID\_REGS Registers

Table 3-352 lists the UID\_REGS registers. All register offset addresses not listed in Table 3-352 should be considered as reserved locations and the register contents should not be modified.

**Table 3-352. UID\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	UID_PSRAND0	UID Psuedo-random 192 bit number		<a href="#">Go</a>
2h	UID_PSRAND1	UID Psuedo-random 192 bit number		<a href="#">Go</a>
4h	UID_PSRAND2	UID Psuedo-random 192 bit number		<a href="#">Go</a>
6h	UID_PSRAND3	UID Psuedo-random 192 bit number		<a href="#">Go</a>
8h	UID_PSRAND4	UID Psuedo-random 192 bit number		<a href="#">Go</a>
Ah	UID_PSRAND5	UID Psuedo-random 192 bit number		<a href="#">Go</a>
Ch	UID_UNIQUE	UID Unique 32 bit number		<a href="#">Go</a>
Eh	UID_CHECKSUM	UID Checksum		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-353 shows the codes that are used for access types in this section.

**Table 3-353. UID\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.22.1 UID\_PSRAND0 Register (Offset = 0h) [reset = X]**

UID\_PSRAND0 is shown in [Figure 3-316](#) and described in [Table 3-354](#).

Return to the [Summary Table](#).

UID Psuedo-random 192 bit number

**Figure 3-316. UID\_PSRAND0 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
RandomID																															
R-X																															

**Table 3-354. UID\_PSRAND0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	X	Psuedorandom portion of the UID Reset type: N/A

### 3.14.22.2 UID\_PSRAND1 Register (Offset = 2h) [reset = X]

UID\_PSRAND1 is shown in [Figure 3-317](#) and described in [Table 3-355](#).

Return to the [Summary Table](#).

UID Psuedo-random 192 bit number

**Figure 3-317. UID\_PSRAND1 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
RandomID																															
R-X																															

**Table 3-355. UID\_PSRAND1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	X	Psuedorandom portion of the UID Reset type: N/A

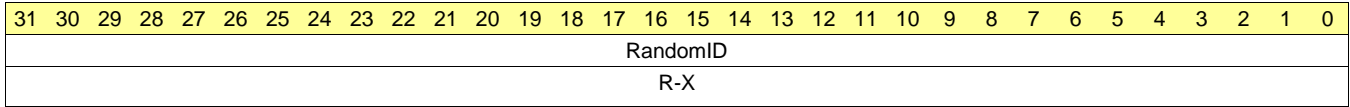
**3.14.22.3 UID\_PSRAND2 Register (Offset = 4h) [reset = X]**

UID\_PSRAND2 is shown in [Figure 3-318](#) and described in [Table 3-356](#).

Return to the [Summary Table](#).

UID Psuedo-random 192 bit number

**Figure 3-318. UID\_PSRAND2 Register**



**Table 3-356. UID\_PSRAND2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	X	Psuedorandom portion of the UID Reset type: N/A

### 3.14.22.4 UID\_PSRAND3 Register (Offset = 6h) [reset = X]

UID\_PSRAND3 is shown in [Figure 3-319](#) and described in [Table 3-357](#).

Return to the [Summary Table](#).

UID Psuedo-random 192 bit number

**Figure 3-319. UID\_PSRAND3 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
RandomID																															
R-X																															

**Table 3-357. UID\_PSRAND3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	X	Psuedorandom portion of the UID Reset type: N/A



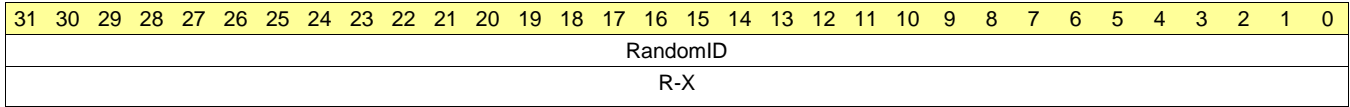
**3.14.22.5 UID\_PSRAND4 Register (Offset = 8h) [reset = X]**

UID\_PSRAND4 is shown in [Figure 3-320](#) and described in [Table 3-358](#).

Return to the [Summary Table](#).

UID Psuedo-random 192 bit number

**Figure 3-320. UID\_PSRAND4 Register**



**Table 3-358. UID\_PSRAND4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	X	Psuedorandom portion of the UID Reset type: N/A

### 3.14.22.6 UID\_PSRAND5 Register (Offset = Ah) [reset = X]

UID\_PSRAND5 is shown in [Figure 3-321](#) and described in [Table 3-359](#).

Return to the [Summary Table](#).

UID Psuedo-random 192 bit number

**Figure 3-321. UID\_PSRAND5 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
RandomID																															
R-X																															

**Table 3-359. UID\_PSRAND5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	X	Psuedorandom portion of the UID Reset type: N/A

**3.14.22.7 UID\_UNIQUE Register (Offset = Ch) [reset = X]**

UID\_UNIQUE is shown in [Figure 3-322](#) and described in [Table 3-360](#).

Return to the [Summary Table](#).

UID Unique 32 bit number

**Figure 3-322. UID\_UNIQUE 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
UniqueID																															
R-X																															

**Table 3-360. UID\_UNIQUE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	UniqueID	R	X	Unique portion of the UID. This identifier will be unique across all devices with the same PARTIDH. Reset type: N/A

### 3.14.22.8 UID\_CHECKSUM Register (Offset = Eh) [reset = X]

UID\_CHECKSUM is shown in [Figure 3-323](#) and described in [Table 3-361](#).

Return to the [Summary Table](#).

Fletcher checksum of UID\_PSRAND and UID\_UNIQUE registers

**Figure 3-323. UID\_CHECKSUM 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
Checksum																															
R-X																															

**Table 3-361. UID\_CHECKSUM Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Checksum	R	X	Fletcher checksum of UID_PSRANDx and UID_UNIQUE Reset type: N/A

### 3.14.23 WD\_REGS Registers

Table 3-362 lists the WD\_REGS registers. All register offset addresses not listed in Table 3-362 should be considered as reserved locations and the register contents should not be modified.

**Table 3-362. WD\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
22h	SCSR	System Control & Status Register	EALLOW	<a href="#">Go</a>
23h	WDCNTR	Watchdog Counter Register	EALLOW	<a href="#">Go</a>
25h	WDKEY	Watchdog Reset Key Register	EALLOW	<a href="#">Go</a>
29h	WDCR	Watchdog Control Register	EALLOW	<a href="#">Go</a>
2Ah	WDWCR	Watchdog Windowed Control Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-363 shows the codes that are used for access types in this section.

**Table 3-363. WD\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.23.1 SCSR Register (Offset = 22h) [reset = 5h]**

SCSR is shown in [Figure 3-324](#) and described in [Table 3-364](#).

Return to the [Summary Table](#).

System Control & Status Register

**Figure 3-324. SCSR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED					WDINTS	WDENINT	WDOVERRIDE
R-0-0h					R-1h	R/W-0h	R/W1S-1h

**Table 3-364. SCSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R-0	0h	Reserved
2	WDINTS	R	1h	Watchdog Interrupt Status  This bit indicates the state of the active-low watchdog interrupt signal (synchronized to SYSCLK). If the watchdog interrupt is used to wake the system from a low-power mode, then that mode should only be entered while this bit is high. Likewise, this bit must go high before the watchdog can be safely disabled and re-enabled.  Reset type: SYSRSn 0h (R/W) = The watchdog interrupt signal is active. 1h (R/W) = The watchdog interrupt signal is inactive.
1	WDENINT	R/W	0h	Watchdog Interrupt Enable/Reset Disable  This bit determines whether the watchdog triggers an interrupt (WAKE/WDOG) or a reset (WDRS) when the counter expires.  Reset type: SYSRSn 0h (R/W) = Counter expiration triggers a reset. This is the default state on power-up and after any system reset. 1h (R/W) = Counter expiration triggers an interrupt.
0	WDOVERRIDE	R/W1S	1h	Watchdog Enable Lock  Writing a 1 to this bit clears it and locks the WDDIS bit in the WDCR register. The bit will remain in this state until the next system reset. Reads of this bit return its current value. Writing a 0 to this bit has no effect.  Reset type: SYSRSn

**3.14.23.2 WDCNTR Register (Offset = 23h) [reset = 0h]**

WDCNTR is shown in [Figure 3-325](#) and described in [Table 3-365](#).

Return to the [Summary Table](#).

Watchdog Counter Register

**Figure 3-325. WDCNTR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
WDCNTR							
R-0h							

**Table 3-365. WDCNTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-0	WDCNTR	R	0h	Watchdog Counter  These bits contain the current value of the watchdog counter. This counter increments with each WDCLK (INTOSC1) cycle. If the counter overflows, either an interrupt or a reset is generated based on the value of the WDINTEN bit in the SCSR register. If the correct value is written to the WDKEY register, this counter is reset to zero.  Reset type: IORSn

### 3.14.23.3 WDKEY Register (Offset = 25h) [reset = 0h]

WDKEY is shown in [Figure 3-326](#) and described in [Table 3-366](#).

Return to the [Summary Table](#).

Watchdog Reset Key Register

**Figure 3-326. WDKEY Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
WDKEY							
R/W-0h							

**Table 3-366. WDKEY Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-0	WDKEY	R/W	0h	Watchdog Counter Reset  Writing 0x55 followed by 0xAA will cause the watchdog counter to reset to zero, preventing an overflow. Writing other values has no effect. Reads of this register return the value of the WDCR register.  Reset type: IORSn



### 3.14.23.4 WDCR Register (Offset = 29h) [reset = 0h]

WDCR is shown in [Figure 3-327](#) and described in [Table 3-367](#).

Return to the [Summary Table](#).

Watchdog Control Register

**Figure 3-327. WDCR Register**

15	14	13	12	11	10	9	8
RESERVED				WDPRECLKDIV			
R-0-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED	WDDIS	WDCHK		WDPS			
R/W-0h		R-0/W-0h		R/W-0h			

**Table 3-367. WDCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-8	WDPRECLKDIV	R/W	0h	<p>Watchdog Clock Pre-divider</p> <p>These bits determine the watchdog clock pre-divider, which is the first of the two dividers between INTOSC1 and the watchdog counter clock (WDCLK). The frequency of WDCLK is given by the formulas:</p> <p>PREDIVCLK = INTOSC1 / Pre-divider WDCLK = PREDIVCLK / Prescaler</p> <p>The watchdog reset or interrupt pulse is 512 INTOSC1 cycles long, so the counter period must be longer. To guarantee this, the product of the prescaler and pre-divider must be greater than or equal to four. The default pre-divider value is 512.</p> <p>Reset type: IORSn</p> <p>0h (R/W) = PREDIVCLK = INTOSC1 / 512 1h (R/W) = PREDIVCLK = INTOSC1 / 1024 2h (R/W) = PREDIVCLK = INTOSC1 / 2048 3h (R/W) = PREDIVCLK = INTOSC1 / 4096 4h (R/W) = Reserved 5h (R/W) = Reserved 6h (R/W) = Reserved 7h (R/W) = Reserved 8h (R/W) = PREDIVCLK = INTOSC1 / 2 9h (R/W) = PREDIVCLK = INTOSC1 / 4 Ah (R/W) = PREDIVCLK = INTOSC1 / 8 Bh (R/W) = PREDIVCLK = INTOSC1 / 16 Ch (R/W) = PREDIVCLK = INTOSC1 / 32 Dh (R/W) = PREDIVCLK = INTOSC1 / 64 Eh (R/W) = PREDIVCLK = INTOSC1 / 128 Fh (R/W) = PREDIVCLK = INTOSC1 / 256</p>
7	RESERVED	R/W1S	0h	Reserved
6	WDDIS	R/W	0h	<p>Watchdog Disable</p> <p>Setting this bit disables the watchdog module. Clearing this bit enables the watchdog module. This bit can be locked by the WDOVERRIDE bit in the SCSR register. The watchdog is enabled on reset.</p> <p>Reset type: IORSn</p>

**Table 3-367. WDCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-3	WDCHK	R-0/W	0h	Watchdog Check Bits  The user must ALWAYS write 1,0,1 to these bits whenever a write to this register is performed. Writing any other value will cause an immediate reset to the core (if WD enabled)  Reset type: IORSn
2-0	WDPS	R/W	0h	Watchdog Clock Prescaler  These bits determine the watchdog clock prescaler, which is the second of the two dividers between INTOSC1 and the watchdog counter clock (WDCLK). The frequency of WDCLK is given by the formulas:  $PREDIVCLK = INTOSC1 / \text{Pre-divider}$ $WDCLK = PREDIVCLK / \text{Prescaler}$  The watchdog reset or interrupt pulse is 512 INTOSC1 cycles long, so the counter period must be longer. To guarantee this, the product of the prescaler and pre-divider must be greater than or equal to four. The default prescaler value is 1.  Reset type: IORSn 0h (R/W) = $WDCLK = PREDIVCLK / 1$ 1h (R/W) = $WDCLK = PREDIVCLK / 1$ 2h (R/W) = $WDCLK = PREDIVCLK / 2$ 3h (R/W) = $WDCLK = PREDIVCLK / 4$ 4h (R/W) = $WDCLK = PREDIVCLK / 8$ 5h (R/W) = $WDCLK = PREDIVCLK / 16$ 6h (R/W) = $WDCLK = PREDIVCLK / 32$ 7h (R/W) = $WDCLK = PREDIVCLK / 64$

### 3.14.23.5 WDWCR Register (Offset = 2Ah) [reset = 0h]

WDWCR is shown in [Figure 3-328](#) and described in [Table 3-368](#).

Return to the [Summary Table](#).

Watchdog Windowed Control Register

**Figure 3-328. WDWCR Register**

15	14	13	12	11	10	9	8
RESERVED							RESERVED
R-0-0h							
7	6	5	4	3	2	1	0
MIN							
R/W-0h							

**Table 3-368. WDWCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R-0	0h	Reserved
8	RESERVED	R	0h	Reserved
7-0	MIN	R/W	0h	Watchdog Window Threshold  These bits specify the lower limit of the watchdog counter reset window. If the counter is reset via the WDKEY register before the counter value reaches the value in this register, the watchdog immediately triggers a reset or interrupt.  Reset type: IORSn

### 3.14.24 XINT\_REGS Registers

Table 3-369 lists the XINT\_REGS registers. All register offset addresses not listed in Table 3-369 should be considered as reserved locations and the register contents should not be modified.

**Table 3-369. XINT\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	XINT1CR	XINT1 configuration register		<a href="#">Go</a>
1h	XINT2CR	XINT2 configuration register		<a href="#">Go</a>
2h	XINT3CR	XINT3 configuration register		<a href="#">Go</a>
3h	XINT4CR	XINT4 configuration register		<a href="#">Go</a>
4h	XINT5CR	XINT5 configuration register		<a href="#">Go</a>
8h	XINT1CTR	XINT1 counter register		<a href="#">Go</a>
9h	XINT2CTR	XINT2 counter register		<a href="#">Go</a>
Ah	XINT3CTR	XINT3 counter register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-370 shows the codes that are used for access types in this section.

**Table 3-370. XINT\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**3.14.24.1 XINT1CR Register (Offset = 0h) [reset = 0h]**

XINT1CR is shown in [Figure 3-329](#) and described in [Table 3-371](#).

Return to the [Summary Table](#).

XINT1 configuration register

**Figure 3-329. XINT1CR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				POLARITY		RESERVED	ENABLE
R-0-0h				R/W-0h		R-0-0h	R/W-0h

**Table 3-371. XINT1CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn

**3.14.24.2 XINT2CR Register (Offset = 1h) [reset = 0h]**

XINT2CR is shown in [Figure 3-330](#) and described in [Table 3-372](#).

Return to the [Summary Table](#).

XINT2 configuration register

**Figure 3-330. XINT2CR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				POLARITY		RESERVED	ENABLE
R-0-0h				R/W-0h		R-0-0h	R/W-0h

**Table 3-372. XINT2CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn

**3.14.24.3 XINT3CR Register (Offset = 2h) [reset = 0h]**

XINT3CR is shown in [Figure 3-331](#) and described in [Table 3-373](#).

Return to the [Summary Table](#).

XINT3 configuration register

**Figure 3-331. XINT3CR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				POLARITY		RESERVED	ENABLE
R-0-0h				R/W-0h		R-0-0h	R/W-0h

**Table 3-373. XINT3CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn

**3.14.24.4 XINT4CR Register (Offset = 3h) [reset = 0h]**

XINT4CR is shown in [Figure 3-332](#) and described in [Table 3-374](#).

Return to the [Summary Table](#).

XINT4 configuration register

**Figure 3-332. XINT4CR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				POLARITY		RESERVED	ENABLE
R-0-0h				R/W-0h		R-0-0h	R/W-0h

**Table 3-374. XINT4CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn



**3.14.24.5 XINT5CR Register (Offset = 4h) [reset = 0h]**

XINT5CR is shown in [Figure 3-333](#) and described in [Table 3-375](#).

Return to the [Summary Table](#).

XINT5 configuration register

**Figure 3-333. XINT5CR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				POLARITY		RESERVED	ENABLE
R-0-0h				R/W-0h		R-0-0h	R/W-0h

**Table 3-375. XINT5CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn

### 3.14.24.6 XINT1CTR Register (Offset = 8h) [reset = 0h]

XINT1CTR is shown in [Figure 3-334](#) and described in [Table 3-376](#).

Return to the [Summary Table](#).

XINT1 counter register

**Figure 3-334. XINT1CTR Register**

15	14	13	12	11	10	9	8
INTCTR							
R-0h							
7	6	5	4	3	2	1	0
INTCTR							
R-0h							

**Table 3-376. XINT1CTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	INTCTR	R	0h	This is a free running 16-bit up-counter that is clocked at the SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. The counter must only be reset by the selected POLARITY edge as selected in the respective interrupt control register. When the interrupt is disabled, the counter will stop. The counter is a free-running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset. Reset type: SYSRSn

### 3.14.24.7 XINT2CTR Register (Offset = 9h) [reset = 0h]

XINT2CTR is shown in [Figure 3-335](#) and described in [Table 3-377](#).

Return to the [Summary Table](#).

XINT2 counter register

**Figure 3-335. XINT2CTR Register**

15	14	13	12	11	10	9	8
INTCTR							
R-0h							
7	6	5	4	3	2	1	0
INTCTR							
R-0h							

**Table 3-377. XINT2CTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	INTCTR	R	0h	<p>This is a free running 16-bit up-counter that is clocked at the SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. The counter must only be reset by the selected POLARITY edge as selected in the respective interrupt control register. When the interrupt is disabled, the counter will stop. The counter is a free-running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset.</p> <p>Reset type: SYSRSn</p>

### 3.14.24.8 XINT3CTR Register (Offset = Ah) [reset = 0h]

XINT3CTR is shown in [Figure 3-336](#) and described in [Table 3-378](#).

Return to the [Summary Table](#).

XINT3 counter register

**Figure 3-336. XINT3CTR Register**

15	14	13	12	11	10	9	8
INTCTR							
R-0h							
7	6	5	4	3	2	1	0
INTCTR							
R-0h							

**Table 3-378. XINT3CTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	INTCTR	R	0h	This is a free running 16-bit up-counter that is clocked at the SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. The counter must only be reset by the selected POLARITY edge as selected in the respective interrupt control register. When the interrupt is disabled, the counter will stop. The counter is a free-running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset.  Reset type: SYSRSn

### 3.14.25 Register to Driverlib Function Mapping

**Table 3-379. CPUTIMER Registers to Driverlib Functions**

File	Driverlib Function
<b>TIM</b>	
cputimer.h	CPUTimer_getTimerCount
<b>PRD</b>	
cputimer.h	CPUTimer_setPeriod
<b>TCR</b>	
cputimer.c	CPUTimer_setEmulationMode
cputimer.h	CPUTimer_clearOverflowFlag
cputimer.h	CPUTimer_disableInterrupt
cputimer.h	CPUTimer_enableInterrupt
cputimer.h	CPUTimer_reloadTimerCounter
cputimer.h	CPUTimer_stopTimer
cputimer.h	CPUTimer_resumeTimer
cputimer.h	CPUTimer_startTimer
cputimer.h	CPUTimer_getTimerOverflowStatus
<b>TPR</b>	
cputimer.h	CPUTimer_setPreScaler
<b>TPRH</b>	
cputimer.h	CPUTimer_setPreScaler

**Table 3-380. ASYSCTL Registers to Driverlib Functions**

File	Driverlib Function
<b>TSNSCTL</b>	
asysctl.h	ASysCtl_enableTemperatureSensor
asysctl.h	ASysCtl_disableTemperatureSensor
<b>ANAREFCTL</b>	
adc.c	ADC_setVREF
asysctl.h	ASysCtl_setAnalogReferenceInternal
asysctl.h	ASysCtl_setAnalogReferenceExternal
asysctl.h	ASysCtl_setAnalogReference2P5
asysctl.h	ASysCtl_setAnalogReference1P65
<b>DCDCCTL</b>	
asysctl.h	ASysCtl_enableDCDC
asysctl.h	ASysCtl_disableDCDC
<b>DCDCSTS</b>	
asysctl.h	ASysCtl_getInductorFaultStatus
asysctl.h	ASysCtl_getSwitchSequenceStatus
<b>CMPPMXSEL</b>	
asysctl.h	ASysCtl_selectCMPPMux
<b>CMPLPMXSEL</b>	
asysctl.h	ASysCtl_selectCMPLPMux
<b>CMPHNMXSEL</b>	
asysctl.h	ASysCtl_selectCMPHNMux
<b>CMPLNMXSEL</b>	
asysctl.h	ASysCtl_selectCMPLNMux

**Table 3-380. ASYSCTL Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>LOCK</b>	
asysctl.h	ASysCtl_lockTemperatureSensor
asysctl.h	ASysCtl_lockANAREF
asysctl.h	ASysCtl_lockVMON
asysctl.h	ASysCtl_lockDCDC
asysctl.h	ASysCtl_lockPGAADCINMux
asysctl.h	ASysCtl_lockCMPHPMux
asysctl.h	ASysCtl_lockCMPLPMux
asysctl.h	ASysCtl_lockCMPHNMux
asysctl.h	ASysCtl_lockCMPLNMux
asysctl.h	ASysCtl_lockVREG

**Table 3-381. PIE Registers to Driverlib Functions**

File	Driverlib Function
<b>CTRL</b>	
interrupt.c	Interrupt_initModule
interrupt.h	Interrupt_defaultHandler
<b>ACK</b>	
interrupt.c	Interrupt_disable
interrupt.h	Interrupt_clearACKGroup
<b>IER1</b>	
interrupt.c	Interrupt_initModule
interrupt.c	Interrupt_enable
interrupt.c	Interrupt_disable
<b>IFR1</b>	
interrupt.c	Interrupt_initModule
<b>IER2</b>	
interrupt.c	Interrupt_initModule
<b>IFR2</b>	
interrupt.c	Interrupt_initModule
<b>IER3</b>	
interrupt.c	Interrupt_initModule
<b>IFR3</b>	
interrupt.c	Interrupt_initModule
<b>IER4</b>	
interrupt.c	Interrupt_initModule
<b>IFR4</b>	
interrupt.c	Interrupt_initModule
<b>IER5</b>	
interrupt.c	Interrupt_initModule
<b>IFR5</b>	
interrupt.c	Interrupt_initModule
<b>IER6</b>	
interrupt.c	Interrupt_initModule
<b>IFR6</b>	
interrupt.c	Interrupt_initModule

**Table 3-381. PIE Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>IER7</b>	
interrupt.c	Interrupt_initModule
<b>IFR7</b>	
interrupt.c	Interrupt_initModule
<b>IER8</b>	
interrupt.c	Interrupt_initModule
<b>IFR8</b>	
interrupt.c	Interrupt_initModule
<b>IER9</b>	
interrupt.c	Interrupt_initModule
<b>IFR9</b>	
interrupt.c	Interrupt_initModule
<b>IER10</b>	
interrupt.c	Interrupt_initModule
<b>IFR10</b>	
interrupt.c	Interrupt_initModule
<b>IER11</b>	
interrupt.c	Interrupt_initModule
<b>IFR11</b>	
interrupt.c	Interrupt_initModule
<b>IER12</b>	
interrupt.c	Interrupt_initModule
<b>IFR12</b>	
interrupt.c	Interrupt_initModule

**Table 3-382. SYSCTL Registers to Driverlib Functions**

File	Driverlib Function
<b>PARTIDL</b>	
sysctl.c	SysCtl_getDeviceParametric
<b>PARTIDH</b>	
sysctl.c	SysCtl_getDeviceParametric
<b>REVID</b>	
sysctl.h	SysCtl_getDeviceRevision
<b>SOFTPRES0</b>	
sysctl.h	SysCtl_resetPeripheral
<b>SOFTPRES2</b>	
-	See SOFTPRES0
<b>SOFTPRES3</b>	
-	See SOFTPRES0
<b>SOFTPRES4</b>	
-	See SOFTPRES0
<b>SOFTPRES6</b>	
-	See SOFTPRES0
<b>SOFTPRES7</b>	
-	See SOFTPRES0

**Table 3-382. SYSTL Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>SOFTPRES8</b>	
-	See SOFTPRES0
<b>SOFTPRES9</b>	
-	See SOFTPRES0
<b>SOFTPRES10</b>	
-	See SOFTPRES0
<b>SOFTPRES13</b>	
-	See SOFTPRES0
<b>SOFTPRES14</b>	
-	See SOFTPRES0
<b>SOFTPRES15</b>	
-	See SOFTPRES0
<b>SOFTPRES16</b>	
-	See SOFTPRES0
<b>CLKSRCCTL1</b>	
sysctl.c	SysCtl_getClock
sysctl.c	SysCtl_selectXTAL
sysctl.c	SysCtl_selectXTALSingleEnded
sysctl.c	SysCtl_selectOscSource
sysctl.h	SysCtl_turnOnOsc
sysctl.h	SysCtl_turnOffOsc
sysctl.h	SysCtl_enableWatchdogInHalt
sysctl.h	SysCtl_disableWatchdogInHalt
<b>CLKSRCCTL2</b>	
can.h	CAN_selectClockSource
<b>CLKSRCCTL3</b>	
sysctl.h	SysCtl_selectClockOutSource
<b>SYSPLLCTL1</b>	
sysctl.c	SysCtl_getClock
sysctl.c	SysCtl_setClock
sysctl.h	SysCtl_enterHaltMode
<b>SYSPLLMULT</b>	
sysctl.c	SysCtl_getClock
sysctl.c	SysCtl_setClock
<b>SYSPLLSTS</b>	
sysctl.c	SysCtl_setClock
<b>SYSCLKDIVSEL</b>	
sysctl.c	SysCtl_getClock
sysctl.c	SysCtl_setClock
<b>LOSPCP</b>	
sysctl.c	SysCtl_getLowSpeedClock
sysctl.h	SysCtl_setLowSpeedClock
<b>MCD CR</b>	
sysctl.h	SysCtl_enableMCD
sysctl.h	SysCtl_disableMCD
sysctl.h	SysCtl_isMCDClockFailureDetected
sysctl.h	SysCtl_resetMCD



**Table 3-382. SYSCTL Registers to Driverlib Functions (continued)**

File	Driverlib Function
sysctl.h	SysCtl_connectMCDClockSource
sysctl.h	SysCtl_disconnectMCDClockSource
<b>X1CNT</b>	
sysctl.c	SysCtl_pollX1Counter
sysctl.h	SysCtl_getExternalOscCounterValue
sysctl.h	SysCtl_clearExternalOscCounterValue
<b>XTALCR</b>	
sysctl.c	SysCtl_selectXTAL
sysctl.c	SysCtl_selectXTALSingleEnded
sysctl.h	SysCtl_setExternalOscMode
sysctl.h	SysCtl_turnOnOsc
sysctl.h	SysCtl_turnOffOsc
<b>PCLKCR0</b>	
sysctl.h	SysCtl_enablePeripheral
sysctl.h	SysCtl_disablePeripheral
<b>PCLKCR2</b>	
-	See PCLKCR0
<b>PCLKCR3</b>	
-	See PCLKCR0
<b>PCLKCR4</b>	
-	See PCLKCR0
<b>PCLKCR6</b>	
-	See PCLKCR0
<b>PCLKCR7</b>	
-	See PCLKCR0
<b>PCLKCR8</b>	
-	See PCLKCR0
<b>PCLKCR9</b>	
-	See PCLKCR0
<b>PCLKCR10</b>	
-	See PCLKCR0
<b>PCLKCR13</b>	
-	See PCLKCR0
<b>PCLKCR14</b>	
-	See PCLKCR0
<b>PCLKCR15</b>	
-	See PCLKCR0
<b>PCLKCR16</b>	
-	See PCLKCR0
<b>LPMCR</b>	
sysctl.h	SysCtl_enterIdleMode
sysctl.h	SysCtl_enterHaltMode
<b>GPIOLPMSEL0</b>	
sysctl.h	SysCtl_enableLPMWakeupPin
sysctl.h	SysCtl_disableLPMWakeupPin
<b>GPIOLPMSEL1</b>	
sysctl.h	SysCtl_enableLPMWakeupPin

**Table 3-382. SYSCCTL Registers to Driverlib Functions (continued)**

File	Driverlib Function
sysctl.h	SysCtl_disableLPMWakeupPin
<b>TMR2CLKCTL</b>	
cputimer.h	CPUTimer_selectClockSource
<b>RESCCLR</b>	
sysctl.h	SysCtl_clearResetCause
sysctl.h	SysCtl_clearWatchdogResetStatus
<b>RESC</b>	
sysctl.h	SysCtl_getResetCause
sysctl.h	SysCtl_clearResetCause
sysctl.h	SysCtl_getWatchdogResetStatus
sysctl.h	SysCtl_clearWatchdogResetStatus
<b>SCSR</b>	
sysctl.h	SysCtl_setWatchdogMode
sysctl.h	SysCtl_isWatchdogInterruptActive
<b>WDCNTR</b>	
sysctl.h	SysCtl_getWatchdogCounterValue
<b>WDKEY</b>	
sysctl.h	SysCtl_serviceWatchdog
<b>WDCR</b>	
sysctl.h	SysCtl_resetDevice
sysctl.h	SysCtl_disableWatchdog
sysctl.h	SysCtl_enableWatchdog
sysctl.h	SysCtl_setWatchdogPredivider
sysctl.h	SysCtl_setWatchdogPrescaler
<b>WDWCR</b>	
sysctl.h	SysCtl_setWatchdogWindowValue
<b>CLA1TASKSRCSEL1</b>	
cla.c	CLA_setTriggerSource
<b>CLA1TASKSRCSEL2</b>	
cla.c	CLA_setTriggerSource
<b>DMACHSRCSEL1</b>	
dma.c	DMA_configMode
<b>DMACHSRCSEL2</b>	
dma.c	DMA_configMode
<b>PERIPH_AC_LOCK</b>	
sysctl.h	SysCtl_lockAccessControlRegs
<b>SYNCSELECT</b>	
sysctl.h	SysCtl_setSyncInputConfig
sysctl.h	SysCtl_setSyncOutputConfig
<b>ADCSOCOUTSELECT</b>	
sysctl.h	SysCtl_enableExtADCSOCSource
sysctl.h	SysCtl_disableExtADCSOCSource
<b>SYNCSOCLOCK</b>	
sysctl.h	SysCtl_lockExtADCSOCSelect
sysctl.h	SysCtl_lockSyncSelect

**Table 3-383. NMI Registers to Driverlib Functions**

File	Driverlib Function
<b>CFG</b>	
sysctl.h	SysCtl_enableNMIGlobalInterrupt
<b>FLG</b>	
sysctl.h	SysCtl_getNMIStatus
sysctl.h	SysCtl_getNMIFlagStatus
sysctl.h	SysCtl_isNMIFlagSet
sysctl.h	SysCtl_clearNMIStatus
sysctl.h	SysCtl_clearAllNMIFlags
sysctl.h	SysCtl_forceNMIFlags
<b>FLGCLR</b>	
sysctl.h	SysCtl_clearNMIStatus
sysctl.h	SysCtl_clearAllNMIFlags
<b>FLGFRC</b>	
sysctl.h	SysCtl_forceNMIFlags
<b>WDCNT</b>	
sysctl.h	SysCtl_getNMIWatchdogCounter
<b>WDPRD</b>	
sysctl.h	SysCtl_setNMIWatchdogPeriod
sysctl.h	SysCtl_getNMIWatchdogPeriod
<b>SHDFLG</b>	
sysctl.h	SysCtl_getNMIShadowFlagStatus
sysctl.h	SysCtl_isNMIShadowFlagSet

**Table 3-384. XINT Registers to Driverlib Functions**

File	Driverlib Function
<b>1CR</b>	
gpio.c	GPIO_setInterruptPin
gpio.h	GPIO_setInterruptType
gpio.h	GPIO_getInterruptType
gpio.h	GPIO_enableInterrupt
gpio.h	GPIO_disableInterrupt
<b>2CR</b>	
-	See 1CR
<b>3CR</b>	
-	See 1CR
<b>4CR</b>	
-	See 1CR
<b>5CR</b>	
-	See 1CR

**Table 3-385. DCSM Registers to Driverlib Functions**

File	Driverlib Function
<b>B0_Z1_LINKPOINTER</b>	
dcsm.c	DCSM_unlockZone1CSM
dcsm.h	DCSM_getZone1LinkPointerError

**Table 3-385. DCSM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>B0_Z1_LINKPOINTERERR</b>	
dcsm.h	DCSM_getZone1LinkPointerError
<b>Z1_CSMKEY0</b>	
dcsm.c	DCSM_unlockZone1CSM
<b>Z1_CSMKEY1</b>	
dcsm.c	DCSM_unlockZone1CSM
<b>Z1_CSMKEY2</b>	
dcsm.c	DCSM_unlockZone1CSM
<b>Z1_CSMKEY3</b>	
dcsm.c	DCSM_unlockZone1CSM
<b>Z1_CR</b>	
dcsm.h	DCSM_secureZone1
dcsm.h	DCSM_getZone1CSMSecurityStatus
dcsm.h	DCSM_getZone1ControlStatus
<b>B0_Z1_EXEONLYSECTR</b>	
dcsm.c	DCSM_getZone1FlashEXEStatus
<b>Z1_EXEONLYRAMR</b>	
dcsm.c	DCSM_getZone1RAMEXEStatus
<b>B0_Z2_LINKPOINTER</b>	
dcsm.c	DCSM_unlockZone2CSM
dcsm.h	DCSM_getZone2LinkPointerError
<b>B0_Z2_LINKPOINTERERR</b>	
dcsm.h	DCSM_getZone2LinkPointerError
<b>Z2_CSMKEY0</b>	
dcsm.c	DCSM_unlockZone2CSM
<b>Z2_CSMKEY1</b>	
dcsm.c	DCSM_unlockZone2CSM
<b>Z2_CSMKEY2</b>	
dcsm.c	DCSM_unlockZone2CSM
<b>Z2_CSMKEY3</b>	
dcsm.c	DCSM_unlockZone2CSM
<b>Z2_CR</b>	
dcsm.h	DCSM_secureZone2
dcsm.h	DCSM_getZone2CSMSecurityStatus
dcsm.h	DCSM_getZone2ControlStatus
<b>B0_Z2_EXEONLYSECTR</b>	
dcsm.c	DCSM_getZone2FlashEXEStatus
<b>Z2_EXEONLYRAMR</b>	
dcsm.c	DCSM_getZone2RAMEXEStatus
<b>FLSEM</b>	
dcsm.c	DCSM_claimZoneSemaphore
dcsm.c	DCSM_releaseZoneSemaphore
<b>B0_SECTSTAT</b>	
dcsm.h	DCSM_getFlashSectorZone
<b>RAMSTAT</b>	
dcsm.h	DCSM_getRAMZone

**Table 3-385. DCSM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>B1_SECTSTAT</b>	
dcsm.h	DCSM_getFlashSectorZone
<b>SECERRSTAT</b>	
dcsm.h	DCSM_getFlashErrorStatus
<b>SECERRCLR</b>	
dcsm.h	DCSM_clearFlashErrorStatus
<b>SECERRFRC</b>	
dcsm.h	DCSM_forceFlashErrorStatus
<b>B1_Z1_LINKPOINTER</b>	
dcsm.h	DCSM_getZone1LinkPointerError
<b>B1_Z1_LINKPOINTERERR</b>	
dcsm.h	DCSM_getZone1LinkPointerError
<b>B1_Z1_EXEONLYSECTR</b>	
dcsm.c	DCSM_getZone1FlashEXEStatus
<b>B1_Z2_LINKPOINTER</b>	
dcsm.h	DCSM_getZone2LinkPointerError
<b>B1_Z2_LINKPOINTERERR</b>	
dcsm.h	DCSM_getZone2LinkPointerError
<b>B1_Z2_EXEONLYSECTR</b>	
dcsm.c	DCSM_getZone2FlashEXEStatus

**Table 3-386. MEMCFG Registers to Driverlib Functions**

File	Driverlib Function
<b>DXLOCK</b>	
memcfg.c	MemCfg_lockConfig
memcfg.c	MemCfg_unlockConfig
<b>DXCOMMIT</b>	
memcfg.c	MemCfg_commitConfig
<b>DXTEST</b>	
memcfg.c	MemCfg_setTestMode
<b>DXINIT</b>	
memcfg.c	MemCfg_initSections
memcfg.c	MemCfg_getInitStatus
<b>DXINITDONE</b>	
memcfg.c	MemCfg_getInitStatus
<b>LSXLOCK</b>	
memcfg.c	MemCfg_lockConfig
memcfg.c	MemCfg_unlockConfig
<b>LSXCOMMIT</b>	
memcfg.c	MemCfg_commitConfig
<b>LSXMSEL</b>	
memcfg.c	MemCfg_setLSRAMMasterSel
<b>LSXCLAPGM</b>	
memcfg.h	MemCfg_setCLAMemType
<b>LSXACCPROTO</b>	
memcfg.c	MemCfg_setProtection

**Table 3-386. MEMCFG Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>LSXTEST</b>	
memcfg.c	MemCfg_setTestMode
<b>LSXINIT</b>	
memcfg.c	MemCfg_initSections
memcfg.c	MemCfg_getInitStatus
<b>LSXINITDONE</b>	
memcfg.c	MemCfg_getInitStatus
<b>GSXLOCK</b>	
memcfg.c	MemCfg_lockConfig
memcfg.c	MemCfg_unlockConfig
<b>GSXCOMMIT</b>	
memcfg.c	MemCfg_commitConfig
<b>GSXACCPROTO</b>	
memcfg.c	MemCfg_setProtection
<b>GSXTEST</b>	
memcfg.c	MemCfg_setTestMode
<b>GSXINIT</b>	
memcfg.c	MemCfg_initSections
memcfg.c	MemCfg_getInitStatus
<b>GSXINITDONE</b>	
memcfg.c	MemCfg_getInitStatus
<b>MSGXLOCK</b>	
memcfg.c	MemCfg_lockConfig
memcfg.c	MemCfg_unlockConfig
<b>MSGXCOMMIT</b>	
memcfg.c	MemCfg_commitConfig
<b>MSGXTEST</b>	
memcfg.c	MemCfg_setTestMode
<b>MSGXINIT</b>	
memcfg.c	MemCfg_initSections
memcfg.c	MemCfg_getInitStatus
<b>MSGXINITDONE</b>	
memcfg.c	MemCfg_getInitStatus
<b>NMAVFLG</b>	
memcfg.h	MemCfg_getViolationInterruptStatus
<b>NMAVSET</b>	
memcfg.h	MemCfg_forceViolationInterrupt
<b>NMAVCLR</b>	
memcfg.h	MemCfg_clearViolationInterruptStatus
<b>NMAVINTEN</b>	
memcfg.h	MemCfg_enableViolationInterrupt
memcfg.h	MemCfg_disableViolationInterrupt
<b>NMCPURDAVADDR</b>	
memcfg.c	MemCfg_getViolationAddress
<b>NMCPUWRAVADDR</b>	
memcfg.c	MemCfg_getViolationAddress

**Table 3-386. MEMCFG Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>MAVFLG</b>	
memcfg.h	MemCfg_getViolationInterruptStatus
<b>MAVSET</b>	
memcfg.h	MemCfg_forceViolationInterrupt
<b>MAVCLR</b>	
memcfg.h	MemCfg_clearViolationInterruptStatus
<b>MAVINTEN</b>	
memcfg.h	MemCfg_enableViolationInterrupt
memcfg.h	MemCfg_disableViolationInterrupt
<b>MCPUFAVADDR</b>	
memcfg.c	MemCfg_getViolationAddress
<b>UCERRFLG</b>	
memcfg.h	MemCfg_getUncorrErrorStatus
<b>UCERRSET</b>	
memcfg.h	MemCfg_forceUncorrErrorStatus
<b>UCERRCLR</b>	
memcfg.h	MemCfg_clearUncorrErrorStatus
<b>UCCPUREADDR</b>	
memcfg.c	MemCfg_getUncorrErrorAddress
<b>UCDMAREADDR</b>	
memcfg.c	MemCfg_getUncorrErrorAddress
<b>CERRFLG</b>	
memcfg.h	MemCfg_getCorrErrorStatus
<b>CERRSET</b>	
memcfg.h	MemCfg_forceCorrErrorStatus
<b>CERRCLR</b>	
memcfg.h	MemCfg_clearCorrErrorStatus
<b>CCPUREADDR</b>	
memcfg.c	MemCfg_getCorrErrorAddress
<b>CERRCNT</b>	
memcfg.h	MemCfg_getCorrErrorCount
<b>CERRTHRES</b>	
memcfg.h	MemCfg_setCorrErrorThreshold
<b>CEINTFLG</b>	
memcfg.h	MemCfg_getCorrErrorInterruptStatus
<b>CEINTCLR</b>	
memcfg.h	MemCfg_clearCorrErrorInterruptStatus
<b>CEINTSET</b>	
memcfg.h	MemCfg_forceCorrErrorInterrupt
<b>CEINTEN</b>	
memcfg.h	MemCfg_enableCorrErrorInterrupt
memcfg.h	MemCfg_disableCorrErrorInterrupt

## ***ROM Code and Peripheral Booting***

---



---

This chapter explains the boot procedure, the available boot modes, and the various details of the ROM code including memory maps, initializations, reset handling, and status information.

Further information about this device can be found in the following document(s):

- [TMS320F28004x Boot Features and Configurations](#)

Topic	Page
<b>4.1 Introduction .....</b>	<b>609</b>
<b>4.2 Device Boot Philosophy .....</b>	<b>609</b>
<b>4.3 Device Boot Modes .....</b>	<b>609</b>
<b>4.4 Device Boot Flow Diagrams .....</b>	<b>614</b>
<b>4.5 Device Reset and Exception Handling .....</b>	<b>617</b>
<b>4.6 Boot ROM Description .....</b>	<b>618</b>
<b>4.7 Application Notes for Using the Bootloaders .....</b>	<b>640</b>



## 4.1 Introduction

The purpose of this chapter is to explain the boot ROM code functionality including the boot procedure when executed, the functions and features of the boot ROM code, and to detail the ROM memory map contents. On every reset, the device executes a boot sequence in the ROM depending on the reset type and boot configuration. This sequence will initialize the device to run application code. The boot ROM also contains peripheral bootloaders which can be used to load an application into RAM.

**Table 4-1. ROM Memory**

ROM	SIZE
Unsecure boot ROM	128 KB
Secure ROM	64 KB
CLA Data ROM	8 KB

## 4.2 Device Boot Philosophy

The boot philosophy describes the general boot ROM procedure each time the CPU core is reset. During booting, the boot ROM code updates a boot status location in RAM that details the actions taken during this process.

Refer to [Boot Status Information](#) for more details on the boot status information.

**Table 4-2. Boot ROM Philosophy**

Step	Action
1	After reset, the FUSE error register is checked for any errors and are handled accordingly.
2	Clock and Flash configuration.
3	Device configuration registers are programmed from OTP.
4	On power-on reset (POR), all CPU RAMs are initialized. RAM initialization includes the following RAMs: <ul style="list-style-type: none"> <li>• M0,M1</li> <li>• LS0 to LS7</li> <li>• GS0 to GS3</li> <li>• CLA1TOCPUMSGRAM, CPUTOCLA1MSGRAM</li> </ul>
5	Any pending NMI is handled by the code.
6	DCSM and OTPJTAGLOCK sequences are executed. (Refer to <a href="#">DCSM Usage</a> for details on how boot ROM interprets the OTP data after initialization)
7	Device calibration is performed, trimming the specified peripherals with set OTP values.
8	The boot mode GPIO pins are polled to determine whether to boot from SRAM, flash, or peripherals.
9	Based on the boot mode and options, the appropriate boot sequence is executed. Refer to <a href="#">Device Boot Flow Diagram</a> for a flow diagram of the device boot sequence and the emulation and standalone boot modes.

## 4.3 Device Boot Modes

This section explains the boot modes supported on this device. The boot ROM uses the boot control GPIO pins to determine the boot mode configuration. The device can be configured to boot to RAM, boot to flash, execute a bootloader, or hold in a wait mode.

[Table 4-3](#) shows the default boot mode options. Users have the option to customize the boot modes supported as well as the boot mode select pins.

**Table 4-3. Device Default Boot Modes**

Boot Mode	GPIO24 (Default boot mode select pin 1)	GPIO32 (Default boot mode select pin 0)
Parallel IO	0	0
SCI / Wait boot	0	1
CAN	1	0
Flash	1	1

**Table 4-4. All Available Boot Modes**

Boot Mode Number	Boot Mode
0	Parallel IO
1	SCI / Wait boot
2	CAN
3	Flash
4	Wait
5	RAM
6	SPI Master
7	I2C Master

**NOTE:** All the peripheral boot modes supported use the first instance of the peripheral module (SCIA, SPIA, I2CA, CANA, and so forth). Whenever these boot modes are referred to in this chapter, such as SCI boot, it is actually referring to the first module instance, meaning SCI boot on the SCIA port. The same applies to the other peripheral boots.

### 4.3.1 Configuring Alternate Boot Mode Pins

This section explains how the boot mode select pins can be customized by the user, by programming the BOOTPIN\_CONFIG location in user-configurable DCSM OTP. The location in user DCSM OTP is Z1-OTP-BOOTPIN-CONFIG. When debugging, EMU-BOOTPIN-CONFIG is the emulation equivalent of Z1-OTP-BOOTPIN-CONFIG, and can be programmed to experiment with different boot modes without writing to OTP. The device can be programmed to use 0, 1, 2, or 3 boot mode select pins as needed.

**Table 4-5. BOOTPIN\_CONFIG Bit Fields**

Bit	Name	Description
31-24	Key	Write 0x5A to these 8-bits to tell the boot ROM code that the bits in this register are valid
23-16	Boot Mode Select Pin 2 (BMSP2)	Refer to BMSP0 description except for BMSP2
15-8	Boot Mode Select Pin 1 (BMSP1)	Refer to BMSP0 description except for BMSP1
7-0	Boot Mode Select Pin 0 (BMSP0)	Set to the GPIO pin to be used during boot (up to 255). 0x0 = GPIO0; 0x01 = GPIO1 and so on 0xFF is invalid and selects the factory default chosen BMSP0, if all other BMSPs are also set to 0xFF. If any other BMSPs are not set to 0xFF, then setting a BMSP to 0xFF will disable that particular BMSP.

**NOTE:** The following GPIOs **cannot** be used as a BMSP. If selected for a particular BMSP, the boot ROM automatically selects the factory default GPIO (the factory default for BMSP2 is 0xFF, which disables the BMSP).

- GPIO 20 to 23
- GPIO 36
- GPIO 38
- GPIO 60 to 223

**Table 4-6. Standalone Boot Mode Select Pin Decoding**

BOOTPIN_CONFIG Key	BMSP0	BMSP1	BMSP2	Realized Boot Mode
!= 0x5A	Don't Care	Don't Care	Don't Care	Boot as defined by the factory default BMSPs (GPIO24, GPIO32)
= 0x5A	0xFF	0xFF	0xFF	Boot as defined in the boot table for boot mode 0 (All BMSPs disabled)
	Valid GPIO	0xFF	0xFF	Boot as defined by the value of BMSP0 (BMSP1 and BMSP2 disabled)
	0xFF	Valid GPIO	0xFF	Boot as defined by the value of BMSP1 (BMSP0 and BMSP2 disabled)
	0xFF	0xFF	Valid GPIO	Boot as defined by the value of BMSP2 (BMSP0 and BMSP1 disabled)
	Valid GPIO	Valid GPIO	0xFF	Boot as defined by the values of BMSP0 and BMSP1 (BMSP2 disabled)
	Valid GPIO	0xFF	Valid GPIO	Boot as defined by the values of BMSP0 and BMSP2 (BMSP1 disabled)
	0xFF	Valid GPIO	Valid GPIO	Boot as defined by the values of BMSP1 and BMSP2 (BMSP0 disabled)
	Valid GPIO	Valid GPIO	Valid GPIO	Boot as defined by the values of BMSP0, BMSP1, and BMSP2

### 4.3.2 Configuring Alternate Boot Mode Options

This section explains how to configure the boot definition table, BOOTDEF, for the device and the associated boot options. The 64-bit location is located in user-configurable DCSM OTP in the Z1-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH locations. When debugging, EMU-BOOTDEF-LOW and EMU-BOOTDEF-HIGH are the emulation equivalents of Z1-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH, and can be programmed to experiment with different boot mode options without writing to OTP. The range of customization to the boot definition table depends on how many boot mode select pins are being used. Refer to [Boot Mode Example Use Cases](#) for examples on how to use the BOOTPIN\_CONFIG and BOOTDEF values.

**Table 4-7. BOOTDEF Bit Fields**

BOOTDEF Name	Byte Position	Name	Description
BOOT_DEF0	7-0	BOOT_DEF0 Mode/Options	Set the boot mode and boot mode options. This can include changing the GPIOs for a particular boot peripheral or specifying a different flash entry point. Any unsupported boot mode will cause the device to reset. Refer to <a href="#">GPIO Assignments</a> for valid BOOTDEF values.
BOOT_DEF1	15-8	BOOT_DEF1 Mode/Options	Refer to BOOT_DEF0 descriptions
BOOT_DEF2	23-16	BOOT_DEF2 Mode/Options	
BOOT_DEF3	31-24	BOOT_DEF3 Mode/Options	
BOOT_DEF4	39-32	BOOT_DEF4 Mode/Options	
BOOT_DEF5	47-40	BOOT_DEF5 Mode/Options	
BOOT_DEF6	55-48	BOOT_DEF6 Mode/Options	
BOOT_DEF7	63-56	BOOT_DEF7 Mode/Options	

### 4.3.3 Boot Mode Example Use Cases

This section demonstrates some use cases for configuring the boot mode select pins.

#### 4.3.3.1 Zero Boot Mode Select Pins

This use case demonstrates a scenario for an application that does not use any boot mode select pins and always has the device boot to flash.

- Program the BOOTPIN\_CONFIG location in OTP as follows:
  - Set BOOTPIN\_CONFIG.BMSP0 to 0xFF
  - Set BOOTPIN\_CONFIG.BMSP1 to 0xFF
  - Set BOOTPIN\_CONFIG.BMSP2 to 0xFF
  - Set BOOTPIN\_CONFIG.KEY to 0x5A for boot ROM to treat these register bits as valid.
- Program the BOOTDEF location options for the device. This essentially sets up a device-specific boot mode table.
  - Set BOOTDEF.BOOTDEF0 to 0x03 for booting to flash with a boot mode value of 0
  - Optionally: Set BOOTDEF.BOOT\_DEF0\_ALT\_OPTIONS to a different value to switch to one of the available flash entry point alternatives.

**Table 4-8. Zero Boot Pin Boot Table Result**

Boot Mode Number	Boot Mode
0	Flash Boot (0x03)

Refer to [Entry Points](#) for the available alternative entry point addresses.

#### 4.3.3.2 One Boot Mode Select Pin

This use case demonstrates a scenario for an application using one boot mode select pin to select between booting to flash or using CAN boot.

- Program the BOOTPIN\_CONFIG location in OTP as follows:
  - Set BOOTPIN\_CONFIG.BMSP0 to a user specified GPIO, such as 0x0 for GPIO0

- Set BOOTPIN\_CONFIG.BMSP1 to 0xFF
  - Set BOOTPIN\_CONFIG.BMSP2 to 0xFF
  - Set BOOTPIN\_CONFIG.KEY to 0x5A for boot ROM to treat these register bits as valid.
2. Program the BOOTDEF location options for the device. This essentially sets up a device-specific boot mode table.
- Set BOOTDEF.BOOTDEF0 to 0x02 for CAN booting with a boot mode value of 0
  - Set BOOTDEF.BOOTDEF1 to 0x03 for booting to flash with a boot mode value of 1
  - Optionally: Set BOOTDEF.BOOT\_DEF1\_ALT\_OPTIONS to a different value to switch to one of the available flash entry point alternatives.

**Table 4-9. One Boot Pin Boot Table Result**

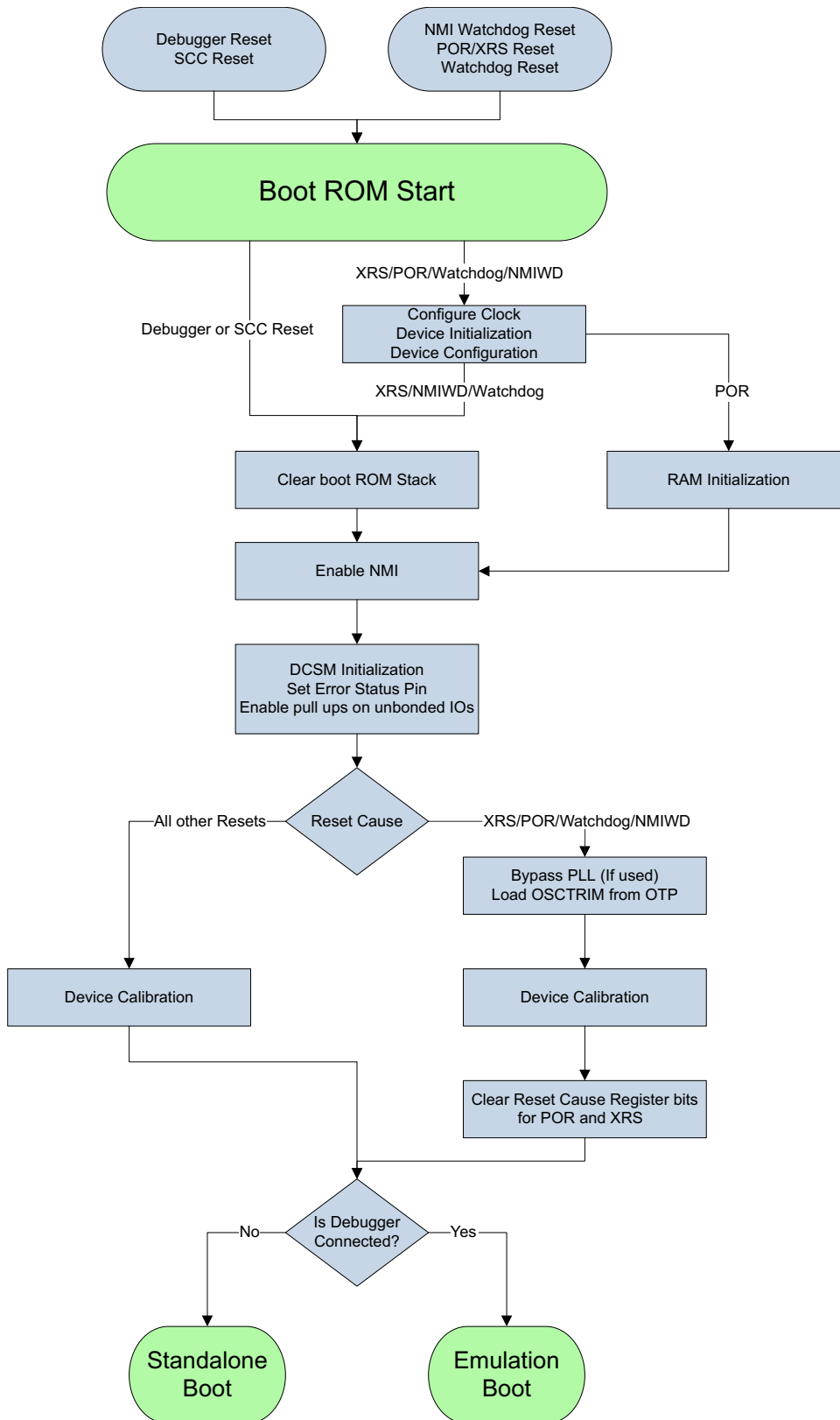
Boot Mode Number	Boot Mode
0	CAN Boot (0x02)
1	Flash Boot (0x03)

Refer to [Entry Points](#) for the available alternative entry point addresses.

### 4.4 Device Boot Flow Diagrams

Figure 4-1 shows the device boot flow detailing the actions executed by boot ROM after a reset.

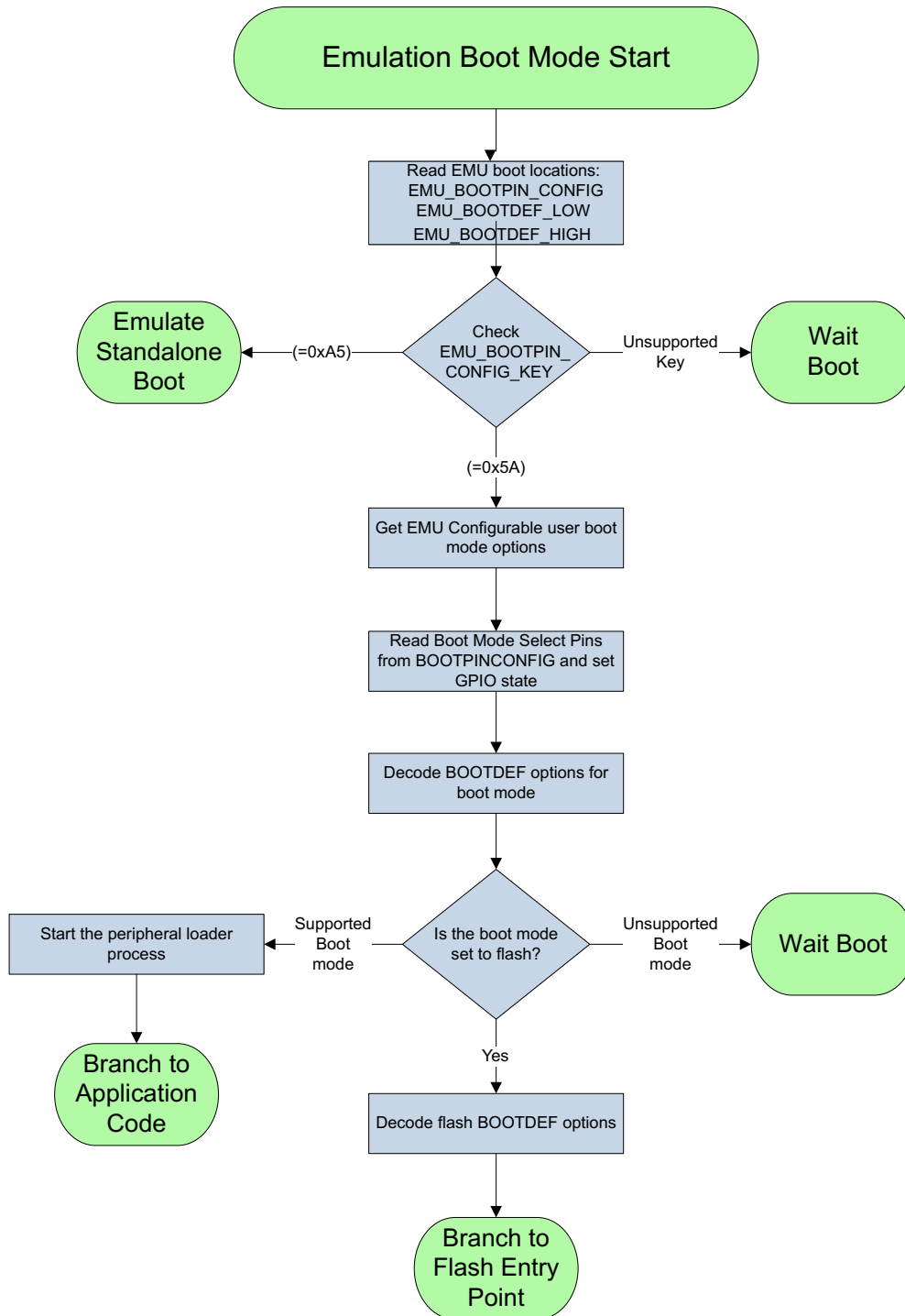
Figure 4-1. Device Boot Flow



### 4.4.1 Emulation Boot Flow Diagram

Figure 4-2 shows the device boot flow when running the device in emulation mode.

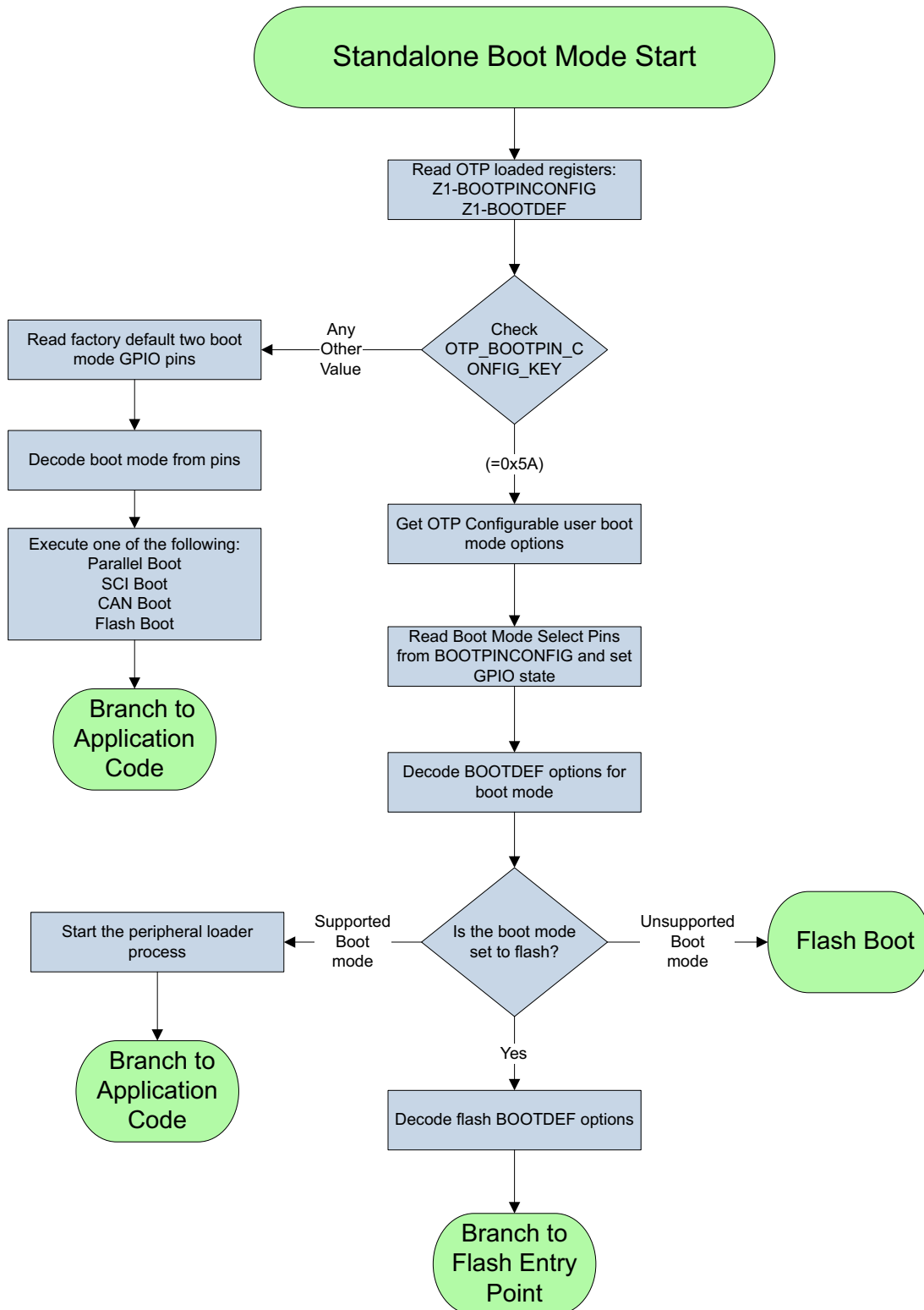
Figure 4-2. Emulation Boot Flow



### 4.4.2 Standalone Boot Flow Diagram

Figure 4-3 shows the device boot flow when running the device in standalone boot mode.

Figure 4-3. Standalone Boot Flow





## 4.5 Device Reset and Exception Handling

### 4.5.1 Reset Causes and Handling

This section explains the actions boot ROM performs upon reset after checking the reset cause.

**Table 4-10. Boot ROM Reset Causes and Actions**

Reset Source	Boot ROM Actions
POR	<ol style="list-style-type: none"> <li>1. Adjust clock divider to /1</li> <li>2. Device configuration</li> <li>3. RAM initialization</li> <li>4. Continue default boot flow</li> </ol>
XRS	<ol style="list-style-type: none"> <li>1. Adjust clock divider to /1</li> <li>2. Device configuration</li> <li>3. Clear boot stack</li> <li>4. Continue default boot flow</li> </ol>
WDRS	<ol style="list-style-type: none"> <li>1. Adjust clock divider to /1</li> <li>2. Device configuration</li> <li>3. Clear boot stack</li> <li>4. Continue default boot flow</li> </ol>
NMIWDRS	<ol style="list-style-type: none"> <li>1. Adjust clock divider to /1</li> <li>2. Device configuration</li> <li>3. Clear boot stack</li> <li>4. Continue default boot flow</li> </ol>
SCCRESET	<ol style="list-style-type: none"> <li>1. Clear boot stack</li> <li>2. Continue default boot flow</li> </ol>
Debugger Reset	<ol style="list-style-type: none"> <li>1. Clear boot stack</li> <li>2. Continue default boot flow</li> </ol>

### 4.5.2 Exceptions and Interrupts Handling

This section explains the actions boot ROM performs if any exceptions that can occur during boot.

**Table 4-11. Boot ROM Exceptions and Actions**

Exception Event Source	Boot ROM Action	Event Logged
Single-bit error in FUSEERR	Ignore and continue to boot	No
Multi-bit error in FUSEERR	Reset the device	No
Clock fail condition detected	Clear NMI, log, and continue to boot	Yes
Double-bit ECC error from RAM	Reset the device	Yes
Double-bit error from Flash	Reset the device	Yes
Software NMI error	Let the device reset	Yes
ITRAP Exception	Give the ROM location where it will loop	Yes

---

**NOTE:** The above NMI errors are logged into a RAM variable for an application to read it when it starts. Refer to [Boot Status Information](#) for more details on the boot status information.

---

## 4.6 Boot ROM Description

This section explains the details regarding the device boot ROM.

### 4.6.1 Boot ROM Registers

The boot ROM code accesses several memory addresses and registers during execution. There are two sets of addresses, one for emulation and one for standalone boot flow. The emulation locations emulate OTP configurations and can be written to as many times as needed. The user configurable DCSM OTP locations used in the standalone boot flow program the device OTP and thus can only be written once. [Table 4-12](#) details these locations.

**Table 4-12. Boot ROM Registers**

Boot Flow	DCSM Name	Boot ROM Name	Address
Emulation	Z1-GPREG1	EMU-BOOTPIN-CONFIG	0x0000 0D00
	Z1-GPREG3	EMU-BOOTDEF-LOW	0x0000 0D04
	Z1-BOOTCTRL	EMU-BOOTDEF-HIGH	0x0000 0D06
Standalone	Z1-GPREG1	Z1-OTP-BOOTPIN-CONFIG	0x0005 F008
	Z1-GPREG2	Z1-OTP-BOOT-GPREG2	0x0005 F00A
	Z1-GPREG3	Z1-OTP-BOOTDEF-LOW	0x0005 F00C
	Z1-BOOTCTRL	Z1-OTP-BOOTDEF-HIGH	0x0005 F004

### 4.6.2 Boot ROM User OTP

The boot ROM user configurable DCSM OTP field descriptions and memory addresses are detailed in [Table 4-13](#).

**Table 4-13. User-Configurable DCSM OTP Fields**

Field name	Description	Bank 0
		USER OTP Address
GPREG1[0:15]	BOOTPIN_CONFIG [0:15]	0x7800C
GPREG1[16:31]	BOOTPIN_CONFIG [16:31]	0x7800D
GPREG2[0:15]	GPREG2[0:15]	0x7800E
GPREG2[16:31]	GPREG2[16:31]	0x7800F
GPREG3[0:15]	BOOTDEF_CONFIG[0:15]	0x7801C
GPREG3[16:31]	BOOTDEF_CONFIG[16:31]	0x7801D
BOOTCTRL[0:15]	BOOTDEF_CONFIG[32:47]	0x7801E
BOOTCTRL[16:31]	BOOTDEF_CONFIG[48:63]	0x7801F

### 4.6.3 Entry Points

This section gives details about the entry point addresses for various boot modes. These entry points tell the boot ROM where to branch to at the end of booting as per the selected boot mode.

**Table 4-14. Entry Point Addresses**

Entry Point	Address
RAM	0x0000 0000
Flash (Option 1 – Default)	0x0008 0000
Flash (Option 2)	0x0008 EFF0
Flash (Option 3) <sup>(1)</sup>	0x0009 0000
Flash (Option 4) <sup>(1)</sup>	0x0009 EFF0

<sup>(1)</sup> This option is not supported on device part numbers that do not have flash bank 1. Please check the device data sheet for the part numbers.

#### 4.6.4 Wait Points

During boot ROM execution, there are situations where the CPU may enter a wait loop in the code. This state can occur for a variety of reasons. [Table 4-15](#) details the address ranges that the CPU PC register value will fall between if it has entered one of these instances.

**Table 4-15. Wait Point Addresses**

Address Range	Description
0x003FAD74 – 0x003FB0CD	In Wait Boot
0x000706DC – 0x000706DF	In SCI Boot
0x003FBCD1 – 0x003FBD67	In NMI Handler
0x003FBD67 – 0x003FBDD5	In ITRAP ISR

#### 4.6.5 Memory Maps

This section details the ROM memory maps.

##### 4.6.5.1 Boot ROM Memory Map

**Table 4-16. Boot ROM Memory Map (Silicon revision 0,A)**

Memory	Start Address	End Address	Length
ROM Signature	0x003F 0000	0x003F 0001	0x0002
TI-RTOS (ROM)	0x003F 7200	0x003F 91FF	0x2000
FPU32 Table	0x003F 9200	0x003F A9FF	0x1800
Boot	0x003F AA00	0x003F E9CF	0x3FD0
Unsecure ROM Checksum	0x003F E9D0	0x003F EA11	0x0042
CPU Spintac Data <sup>(1)</sup>	0x003F EA12	0x003F EA21	0x0010
CPU Fast Data <sup>(1)</sup>	0x003F EA22	0x003F EB21	0x1000
PIE Mismatch Handler	0x003F FF22	0x003F FF71	0x0050
CRC Table	0x003F FF72	0x003F FF79	0x0008
Version	0x003F FF7A	0x003F FF7B	0x0002
Vectors	0x003F FFBE	0x003F FFFF	0x0042
TI-RTOS (Flash)	0x0008 1010	0x0008 13EE	0x3FDF

<sup>(1)</sup> Check the data manual to determine if these are available for your device part number. If not available, treat these sections as reserved.

**Table 4-17. Boot ROM Memory Map (Silicon revision B)**

Memory	Start Address	End Address	Length
ROM Signature	0x003F 0000	0x003F 0001	0x0002
TI-RTOS (ROM)	0x003F 7200	0x003F 91FF	0x2000
FPU32 Tables (1024pt CFFT/RFFT)	0x003F 9200	0x003F 9FF7	0x0DF8
FPU32 Tables (512pt CFFT)	0x003F A200	0x003F A9FF	0x0800
Boot	0x003F AA00	0x003F E9CF	0x3FD0
Unsecure ROM Checksum	0x003F E9D0	0x003F EA11	0x0042
CPU Spintac Data <sup>(1)</sup>	0x003F EA12	0x003F EA21	0x0010
CPU Fast Data <sup>(1)</sup>	0x003F EA22	0x003F EB21	0x1000
Flash API Library (ROM)	0x003F EB22	0x003F F634	0x0B13
Flash API Table	0x003F FEFA	0x003F FF21	0x0028

<sup>(1)</sup> Check the data manual to determine if these are available for your device part number. If not available, treat these sections as reserved.

**Table 4-17. Boot ROM Memory Map (Silicon revision B) (continued)**

Memory	Start Address	End Address	Length
PIE Mismatch Handler	0x003F FF22	0x003F FF71	0x0050
CRC Table	0x003F FF72	0x003F FF79	0x0008
Version	0x003F FF7A	0x003F FF7B	0x0002
Vectors	0x003F FFBE	0x003F FFFF	0x0042
TI-RTOS (Flash)	0x0008 1010	0x0008 13EE	0x03DF

#### 4.6.5.2 CLA Data ROM Memory Map

**Table 4-18. CLA Data ROM Memory Map**

Memory	Start Address	End Address	Length
FFT Tables (Load)	0x0100 1070	0x0100 186F	0x0800
Data (Load)	0x0100 1870	0x0100 1FF9	0x078A
Version (Load)	0x0100 1FFA	0x0100 1FFF	0x0006
FFT Tables (Run)	0x0000 F070	0x0000 F86F	0x0800
Data (Run)	0x0000 F870	0x0000 FFF9	0x078A
Version (Run)	0x0000 FFFA	0x0000 FFFF	0x0006

#### 4.6.5.3 Reserved RAM and Flash Memory Map

This section details memory usage in RAM and Flash that is reserved for boot ROM to use. These memory sections should be reserved in the user application.

**Table 4-19. Reserved RAM and Flash Memory Map (Silicon revision A)**

Memory	Description	Start Address	End Address	Length
RAM	Boot ROM	0x0000 0002	0x0000 00F3	0x00F1
	TI-RTOS <sup>(1)</sup>	0x0000 0780	0x0000 07FF	0x0080
Flash	TI-RTOS <sup>(1)(2)</sup>	0x0008 1010	0x0008 13EE	0x03DF

<sup>(1)</sup> If the user is not planning on using TI-RTOS in ROM, then these memory locations are free to be used by the application.

<sup>(2)</sup> For using the TI-RTOS in flash sector A, TI recommends that this sector be made unsecure, or at minimum, the sector should be verified that there is no secure zone claiming this sector.

**Table 4-20. Reserved RAM and Flash Memory Map (Silicon revision B)**

Memory	Description	Start Address	End Address	Length
RAM	Boot ROM	0x0000 0002	0x0000 00F3	0x00F1
	Flash API Library <sup>(1)</sup>	0x0000 0760	0x0000 077F	0x0020
	TI-RTOS <sup>(1)</sup>	0x0000 0780	0x0000 07FF	0x0080
Flash	TI-RTOS <sup>(2)</sup>	0x0008 1010	0x0008 13EE	0x03DF

<sup>(1)</sup> If the user is not planning on using Flash API or TI-RTOS in ROM, then these memory locations are free to be used by the application.

<sup>(2)</sup> For using the TI-RTOS in flash sector A, TI recommends that this sector be made unsecure, or at minimum, the sector should be verified that there is no secure zone claiming this sector.

## 4.6.6 ROM Tables

This section details the boot ROM and CLA ROM symbol tables.

### 4.6.6.1 Boot ROM Tables

The boot ROM and flash API symbols and their addresses can be located in the .map file included with the released boot ROM source and header code. Within the .map file, locate the Global Symbols category to get a list of the boot ROM symbols and addresses present.

### 4.6.6.2 CLA ROM Tables

This section has the symbol tables included in ROM for the CLA and their addresses.

**Table 4-21. CLA ROM Tables**

Table	Address
_cla_twiddleFactors	0x0000 f070
_cla_bitReversalTable	0x0000 f470
_CLAatan2HalfPITable	0x0000 f870
_CLAINV2PI	0x0000 f874
_CLAatan2Table	0x0000 f876
_CLAasinHalfPITable	0x0000 f9fc
_CLAatan2TableEnd	0x0000 f9fc
_CLAasinTable	0x0000 fa00
_CLAacosinHalfPITable	0x0000 fb86
_CLAasinTableEnd	0x0000 fb86
_CLAacosinTable	0x0000 fb8a
_CLAacosinTableEnd	0x0000 fd0a
_CLAsinTable	0x0000 fd0a
_CLAsincosTable	0x0000 fd0a
_CLAsincosTable_Sin0	0x0000 fd0a
_CLAcosTable	0x0000 fd4a
_CLAsincosTable_Cos0	0x0000 fd4a
_CLAsinTableEnd	0x0000 fe0a
_CLAcosTableEnd	0x0000 fe4c
_CLAsincosTable_TABLE_SIZE	0x0000 fe4c
_CLAsincosTable_TABLE_SIZEDivTwoPi	0x0000 fe4e
_CLAsincosTable_TwoPiDivTABLE_SIZE	0x0000 fe50
_CLAsincosTable_TABLE_MASK	0x0000 fe52
_CLAsincosTable_Coef0	0x0000 fe54
_CLAsincosTable_Coef1	0x0000 fe56
_CLAsincosTable_Coef1_pos	0x0000 fe58
_CLAsincosTable_Coef2	0x0000 fe5a
_CLAsincosTable_Coef3	0x0000 fe5c
_CLAsincosTable_Coef3_neg	0x0000 fe5e
_CLALNV2	0x0000 fe60
_CLAsincosTableEnd	0x0000 fe60
_CLALNVe	0x0000 fe62
_CLALNV10	0x0000 fe64
_CLABIAS	0x0000 fe66
_CLALN_TABLE_MASK1	0x0000 fe68
_CLALN_TABLE_MASK2	0x0000 fe6a

**Table 4-21. CLA ROM Tables (continued)**

Table	Address
_CLALnTable	0x0000 fe6c
_CLAINV1	0x0000 ff32
_CLALnTableEnd	0x0000 ff32
_CLAINV2	0x0000 ff34
_CLAINV3	0x0000 ff36
_CLAINV4	0x0000 ff38
_CLAINV5	0x0000 ff3a
_CLAINV6	0x0000 ff3c
_CLAINV7	0x0000 ff3e
_CLALOG10	0x0000 ff40
_CLAEExpTable	0x0000 ff42
_CLAEExpTableEnd	0x0000 fff4

## 4.6.7 Boot Modes

The available boot modes supported on this device are detailed in this section. Each boot mode allows for various options, providing configurations with different IOs to be used depending on the application.

### 4.6.7.1 Wait Boot Mode

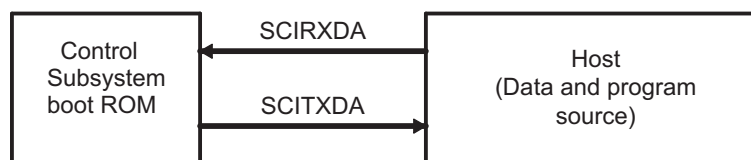
The wait boot mode puts the CPU in a loop and does not branch to the user application code. The device can enter wait boot mode either manually or because an error occurred during boot up. TI recommends using wait boot when using a debugger to avoid any JTAG complications.

Actions resulting in entering wait boot mode:

- Wait boot is set by the user as the boot mode
- The boot mode is unrecognized and a debugger is connected to the device
- The emulation BOOTPIN\_CONFIG key is not equal to 0xA5 or 0x5A
- An error occurs during emulation boot and the boot mode pins are decoded with a value not recognized as a valid boot mode

### 4.6.7.2 SCI Boot Mode

The SCI boot mode asynchronously transfers code from SCI-A to internal memory. This boot mode only supports an incoming 8-bit data stream and follows the data flow as outlined in [Example 4-1](#).

**Figure 4-4. Overview of SCI Bootloader Operation**


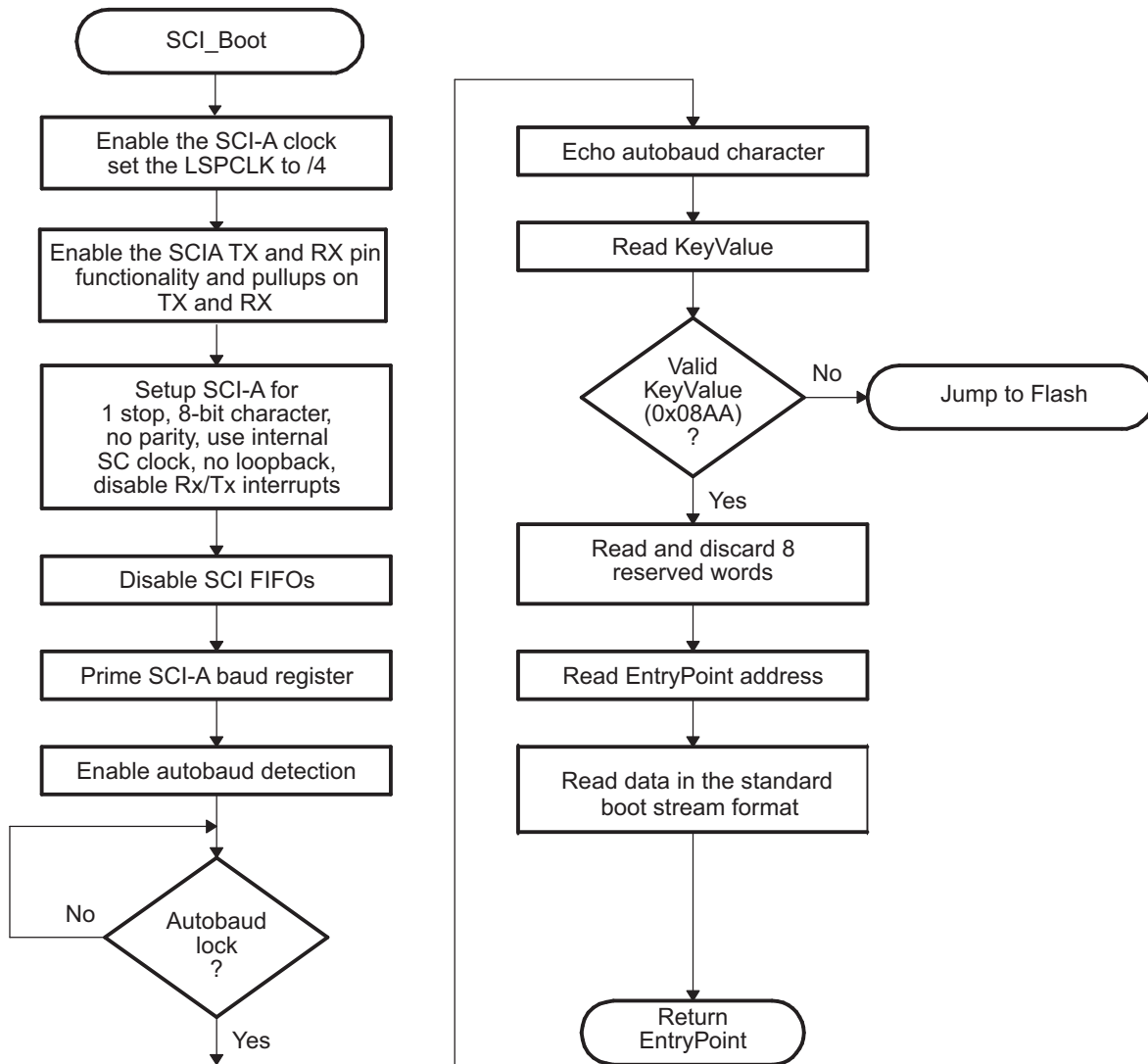
The device communicates with the external host by communication through the SCI-A peripheral. The autobaud feature of the SCI port is used to lock baud rates with the host. For this reason the SCI loader is very flexible and you can use a number of different baud rates to communicate with the device.

After each data transfer, the bootloader will echo back the 8-bit character received to the host. This allows the host to check that each character was received by the bootloader.

At higher baud rates, the slew rate of the incoming data bits can be affected by transceiver and connector performance. While normal serial communications may work well, this slew rate may limit reliable auto-baud detection at higher baud rates (typically beyond 100kbaud) and cause the auto-baud lock feature to fail. To avoid this, the following is recommended:

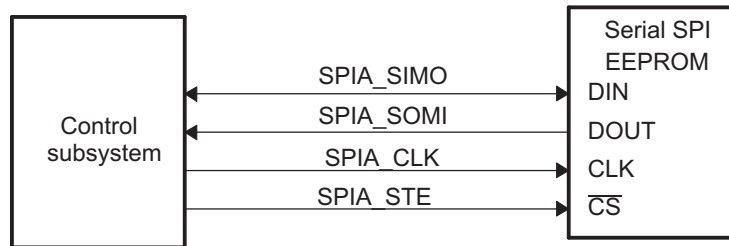
1. Achieve a baud-lock between the host and SCI bootloader using a lower baud rate.
2. Load the incoming application or custom loader at this lower baud rate.
3. The host may then handshake with the loaded application to set the SCI baud rate register to the desired high baud rate.

Figure 4-5. Overview of SCI Boot Function



### 4.6.7.3 SPI Boot Mode

The SPI loader expects an SPI-compatible 16-bit or 24-bit addressable serial EEPROM or serial flash device to be present on the SPI-A pins as indicated in Figure 4-6. The SPI bootloader supports an 8-bit data stream. It does not support a 16-bit data stream.

**Figure 4-6. SPI Loader**


The SPI boot ROM loader initializes the SPI module to interface to a serial SPI EEPROM or flash. Devices of this type include, but are not limited to, the Xicor X25320 (4Kx8) and Xicor X25256 (32Kx8) SPI serial SPI EEPROMs and the Atmel AT25F1024A serial flash.

The SPI boot ROM loader initializes the SPI with the following settings: FIFO enabled, 8-bit character, internal SPICLK master mode and talk mode, clock phase = 1, polarity = 0, using the slowest baud rate.

If the download is to be performed from an SPI port on another device, then that device must be setup to operate in the slave mode and mimic a serial SPI EEPROM. Immediately after entering the SPI\_Boot function, the pin functions for the SPI pins are set to primary and the SPI is initialized. The initialization is done at the slowest speed possible. Once the SPI is initialized and the key value read, you could specify a change in baud rate or low speed peripheral clock.

**Table 4-22. SPI 8-Bit Data Stream**

Byte	Contents
1	LSB: AA (KeyValue for memory width = 8-bits)
2	MSB: 08h (KeyValue for memory width = 8-bits)
3	LSB: LOSPCP
4	MSB: SPIBRR
5	LSB: reserved for future use
6	MSB: reserved for future use
...	...
...	Data for this section.
...	...
17	LSB: reserved for future use
18	MSB: reserved for future use
19	LSB: Upper half (MSW) of Entry point PC[23:16]
20	MSB: Upper half (MSW) of Entry point PC[31:24] (Note: Always 0x00)
21	LSB: Lower half (LSW) of Entry point PC[7:0]
22	MSB: Lower half (LSW) of Entry point PC[15:8]
...	...
...	Data for this section.
...	...
...	Blocks of data in the format size/destination address/data as shown in the generic data stream description
...	...
...	Data for this section.
...	...
n	LSB: 00h
n+1	MSB: 00h - indicates the end of the source

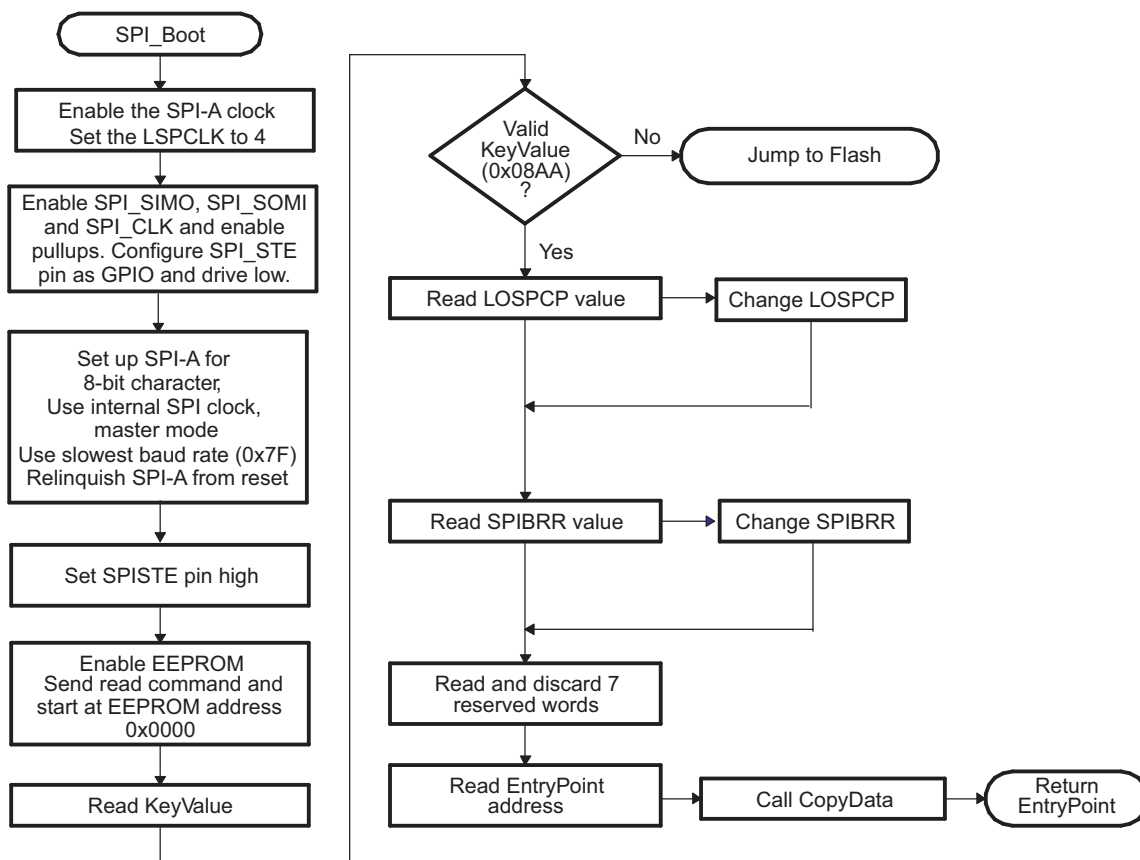
The data transfer is done in "burst" mode from the serial SPI EEPROM. The transfer is carried out entirely in byte mode (SPI at 8 bits/character). A step-by-step description of the sequence follows:

- Step 1. The SPI-A port is initialized
- Step 2. The SPISTE pin is used as a chip-select for the serial SPI EEPROM or flash



- Step 3. The SPI-A outputs a read command for the serial SPI EEPROM or flash
- Step 4. The SPI-A sends the serial SPI EEPROM an address 0x0000; that is, the host requires that the EEPROM or flash must have the downloadable packet starting at address 0x0000 in the EEPROM or flash. The loader is compatible with both 16-bit addresses and 24-bit addresses.
- Step 5. The next word fetched must match the key value for an 8-bit data stream (0x08AA). The least significant byte of this word is the byte read first and the most significant byte is the next byte fetched. This is true of all word transfers on the SPI. If the key value does not match, then the load is aborted and the bootloader jumps to flash.
- Step 6. The next 2 bytes fetched can be used to change the value of the low speed peripheral clock register (LOSPCP) and the SPI baud rate register (SPIBRR). The first byte read is the LOSPCP value and the second byte read is the SPIBRR value. The next 7 words are reserved for future enhancements. The SPI bootloader reads these 7 words and discards them.
- Step 7. The next two words makeup the 32-bit entry point address where execution will continue after the boot load process is complete. This is typically the entry point for the program being downloaded through the SPI port.
- Step 8. Multiple blocks of code and data are then copied into memory from the external serial SPI EEPROM through the SPI port. The blocks of code are organized in the standard data stream structure presented earlier. This is done until a block size of 0x0000 is encountered. At that point in time the entry point address is returned to the calling routine that then exits the bootloader and resumes execution at the address specified.

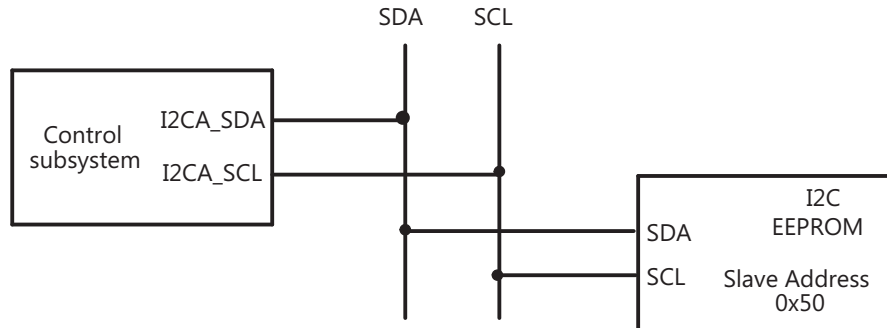
Figure 4-7. Data Transfer From EEPROM Flow



#### 4.6.7.4 I2C Boot Mode

The I2C bootloader expects an 8-bit wide I2C-compatible EEPROM device to be present at address 0x50 on the I2C-A bus as indicated in [Figure 4-8](#). The EEPROM must adhere to conventional I2C EEPROM protocol, as described in this section, with a 16-bit base address architecture.

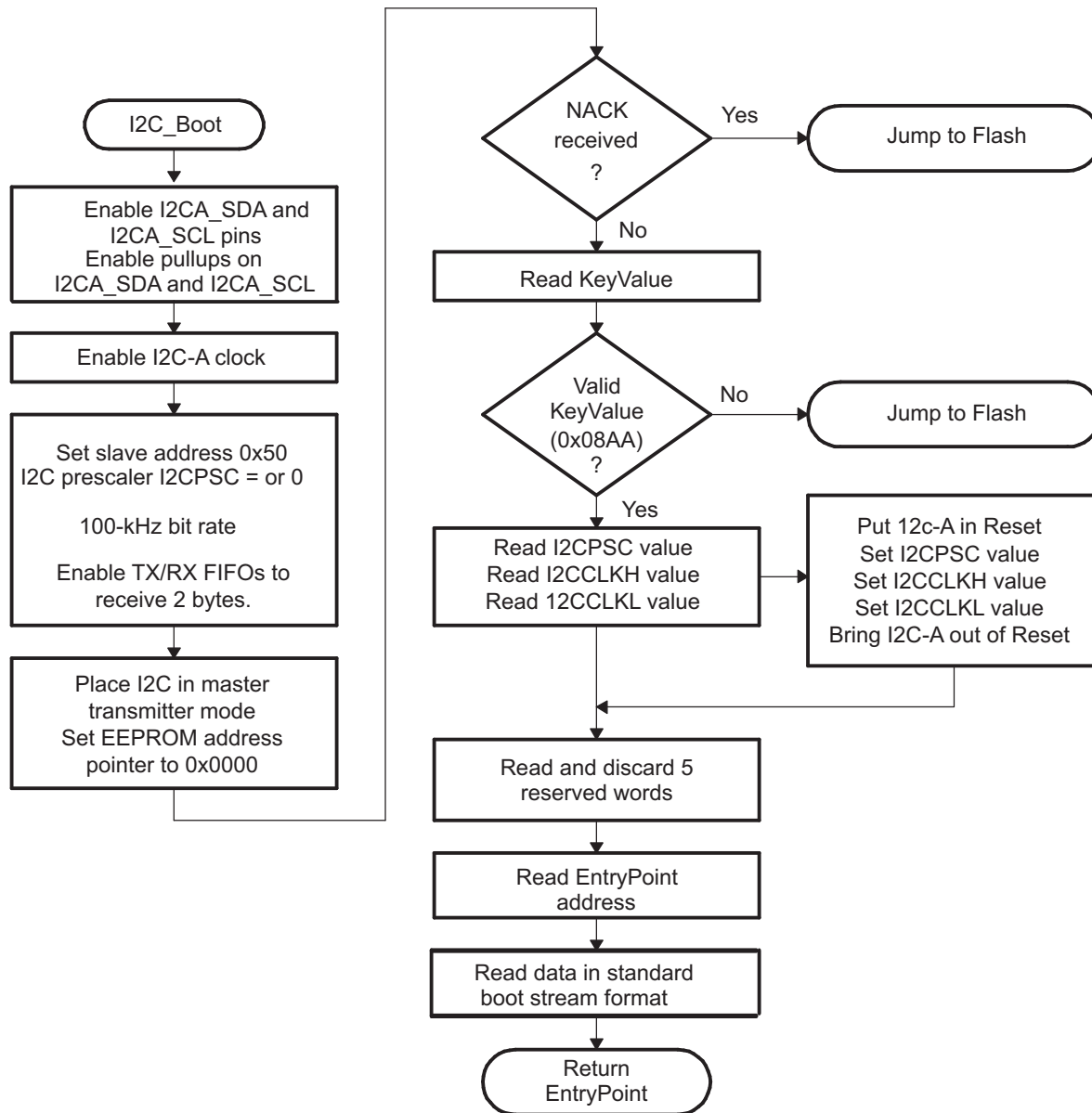
**Figure 4-8. EEPROM Device at Address 0x50**



If the download is to be performed from a device other than an EEPROM, then that device must be set up to operate in the slave mode and mimic the I2C EEPROM. Immediately after entering the I2C boot function, the GPIO pins are configured for I2C-A operation and the I2C is initialized. The following requirements must be met when booting from the I2C module:

- The input frequency to the device must be in the appropriate range.
- The EEPROM must be at slave address 0x50.

Figure 4-9. Overview of I2C Boot Function



The bit-period prescalers (I2CCLKH and I2CCLKL) are configured by the bootloader to run the I2C at a 50 percent duty cycle at 100-kHz bit rate (standard I2C mode) when the system clock is 10 MHz. These registers can be modified after receiving the first few bytes from the EEPROM. This allows the communication to be increased up to a 400-kHz bit rate (fast I2C mode) during the remaining data reads.

Arbitration, bus busy, and slave signals are not checked. Therefore, no other master is allowed to control the bus during this initialization phase. If the application requires another master during I2C boot mode, that master must be configured to hold off sending any I2C messages until the application software signals that it is past the bootloader portion of initialization.

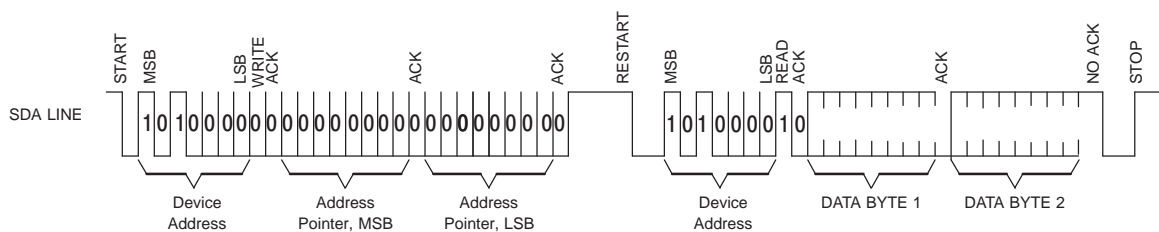
The non-acknowledgment bit is checked only during the first message sent to initialize the EEPROM base address. This is to make sure that an EEPROM is present at address 0x50 before continuing. If an EEPROM is not present, the non-acknowledgment bit is not checked during the address phase of the data read messages (I2C\_Get Word). If a non acknowledgment is received during the data read messages, the I2C bus will hang. [Table 18-1](#) shows the 8-bit data stream used by the I2C.

**Table 4-23. I2C 8-Bit Data Stream**

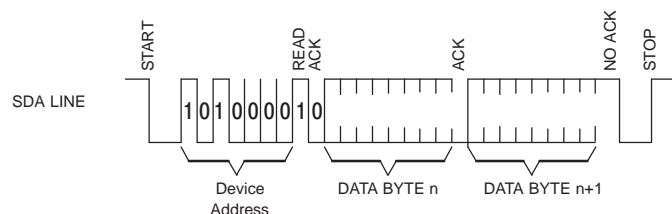
Byte	Contents
1	LSB: AA (KeyValue for memory width = 8 bits)
2	MSB: 08h (KeyValue for memory width = 8 bits)
3	LSB: I2CPSC[7:0]
4	reserved
5	LSB: I2CCLKH[7:0]
6	MSB: I2CCLKH[15:8]
7	LSB: I2CCLKL[7:0]
8	MSB: I2CCLKL[15:8]
...	...
...	Data for this section.
...	...
17	LSB: Reserved for future use
18	MSB: Reserved for future use
19	LSB: Upper half of entry point PC
20	MSB: Upper half of entry point PC[22:16] (Note: Always 0x00)
21	LSB: Lower half of entry point PC[15:8]
22	MSB: Lower half of entry point PC[7:0]
...	...
...	Data for this section.
...	...
...	Blocks of data in the format size/destination address/data as shown in the generic data stream description.
...	...
...	Data for this section.
...	...
n	LSB: 00h
n+1	MSB: 00h - indicates the end of the source

The I2C EEPROM protocol required by the I2C bootloader is shown in [Figure 4-10](#) and [Figure 4-11](#). The first communication, which sets the EEPROM address pointer to 0x0000 and reads the KeyValue (0x08AA) from it, is shown in [Figure 4-10](#). All subsequent reads are shown in [Figure 4-11](#) and are read two bytes at a time.

**Figure 4-10. Random Read**



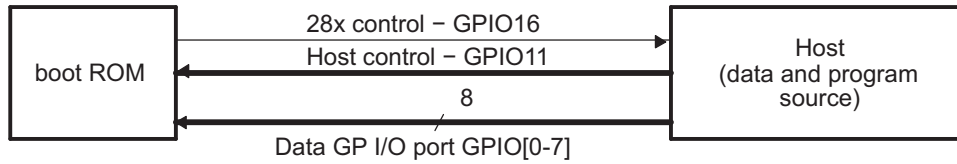
**Figure 4-11. Sequential Read**



### 4.6.7.5 Parallel Boot Mode

The parallel general purpose I/O (GPIO) boot mode asynchronously transfers code from GPIO0 to GPIO7 internal memory. Each value is 8 bits long and follows the same data flow as outlined in [Figure 4-12](#).

**Figure 4-12. Overview of Parallel GPIO Bootloader Operation**



The control subsystem communicates with the external host device by polling/driving the GPIO16 and GPIO11 lines. The handshake protocol shown in [Figure 4-13](#) must be used to successfully transfer each word via GPIO [0-7]. This protocol is very robust and allows for a slower or faster host to communicate with the master subsystem.

Two consecutive 8-bit words are read to form a single 16-bit word. The most significant byte (MSB) is read first followed by the least significant byte (LSB). In this case, data is read from GPIO[0-7].

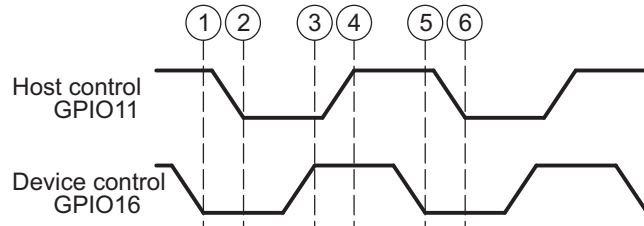
The 8-bit data stream is shown in [Table 4-24](#).

**Table 4-24. Parallel GPIO Boot 8-Bit Data Stream**

Bytes	GPIO[ 7:0] (Byte 1 of 2)	GPIO[ 7:0] (Byte 2 of 2)	Description
1 2	AA	08	0x08AA (KeyValue for memory width = 16bits)
3 4	00	00	8 reserved words (words 2 - 9)
...	...	...	...
17 18	00	00	Last reserved word
19 20	BB	00	Entry point PC[22:16]
21 22	DD	CC	Entry point PC[15:0] (PC = 0x00BBCCDD)
23 24	NN	MM	Block size of the first block of data to load = 0xMMNN words
25 26	BB	AA	Destination address of first block Addr[31:16]
27 28	DD	CC	Destination address of first block Addr[15:0] (Addr = 0xAABCCDD)
29 30	BB	AA	First word of the first block in the source being loaded = 0xAABB
...			...
...			Data for this section.
...			...
.	BB	AA	Last word of the first block of the source being loaded = 0xAABB
.	NN	MM	Block size of the 2nd block to load = 0xMMNN words
.	BB	AA	Destination address of second block Addr[31:16]
.	DD	CC	Destination address of second block Addr[15:0]
.	BB	AA	First word of the second block in the source being loaded
.			...
n n+1	BB	AA	Last word of the last block of the source being loaded (More sections if required)
n+2 n+3	00	00	Block size of 0000h - indicates end of the source program

The device first signals the host that it is ready to begin data transfer by pulling the GPIO16 pin low. The host load then initiates the data transfer by pulling the GPIO11 pin low. The complete protocol is shown in Figure 4-13:

**Figure 4-13. Parallel GPIO Bootloader Handshake Protocol**



1. The device indicates it is ready to start receiving data by pulling the GPIO16 pin low.
2. The bootloader waits until the host puts data on GPIO [0-7]. The host signals to the device that data is ready by pulling the GPIO11 pin low.
3. The device reads the data and signals the host that the read is complete by pulling GPIO16 high.
4. The bootloader waits until the host acknowledges the device by pulling GPIO11 high.
5. The device again indicates it is ready for more data by pulling the GPIO16 pin low.

This process is repeated for each data value to be sent.

Figure 4-14 shows an overview of the Parallel GPIO bootloader flow.

**Figure 4-14. Parallel GPIO Mode Overview**

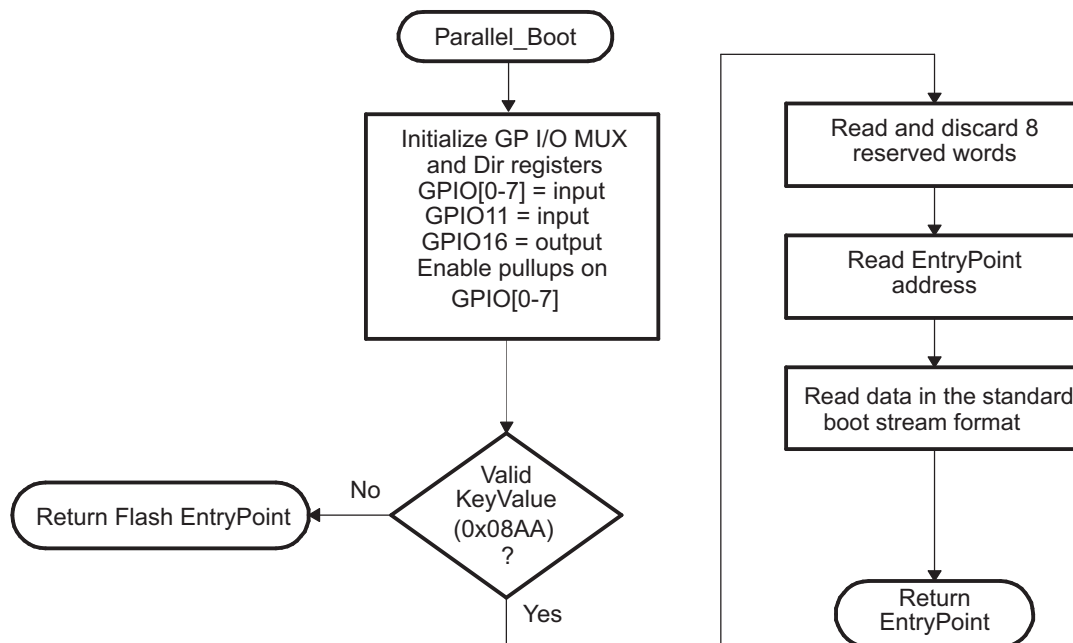


Figure 4-15 shows the transfer flow from the host side. The operating speed of the CPU and host are not critical in this mode as the host will wait for the device and the device will in turn wait for the host. In this manner the protocol will work with both a host running faster and a host running slower than the device.

Figure 4-15. Parallel GPIO Mode - Host Transfer Flow

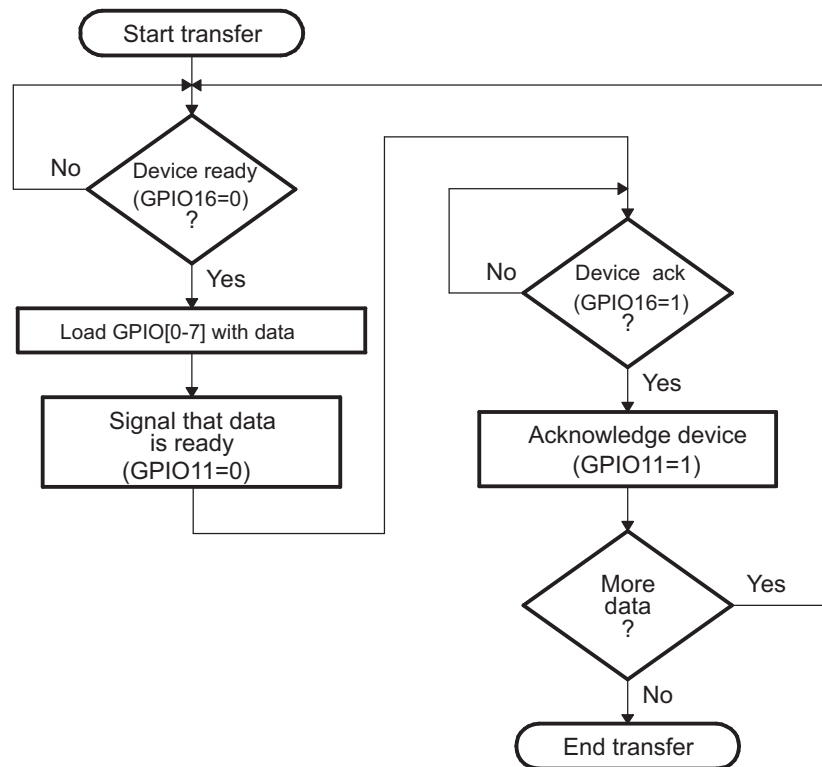
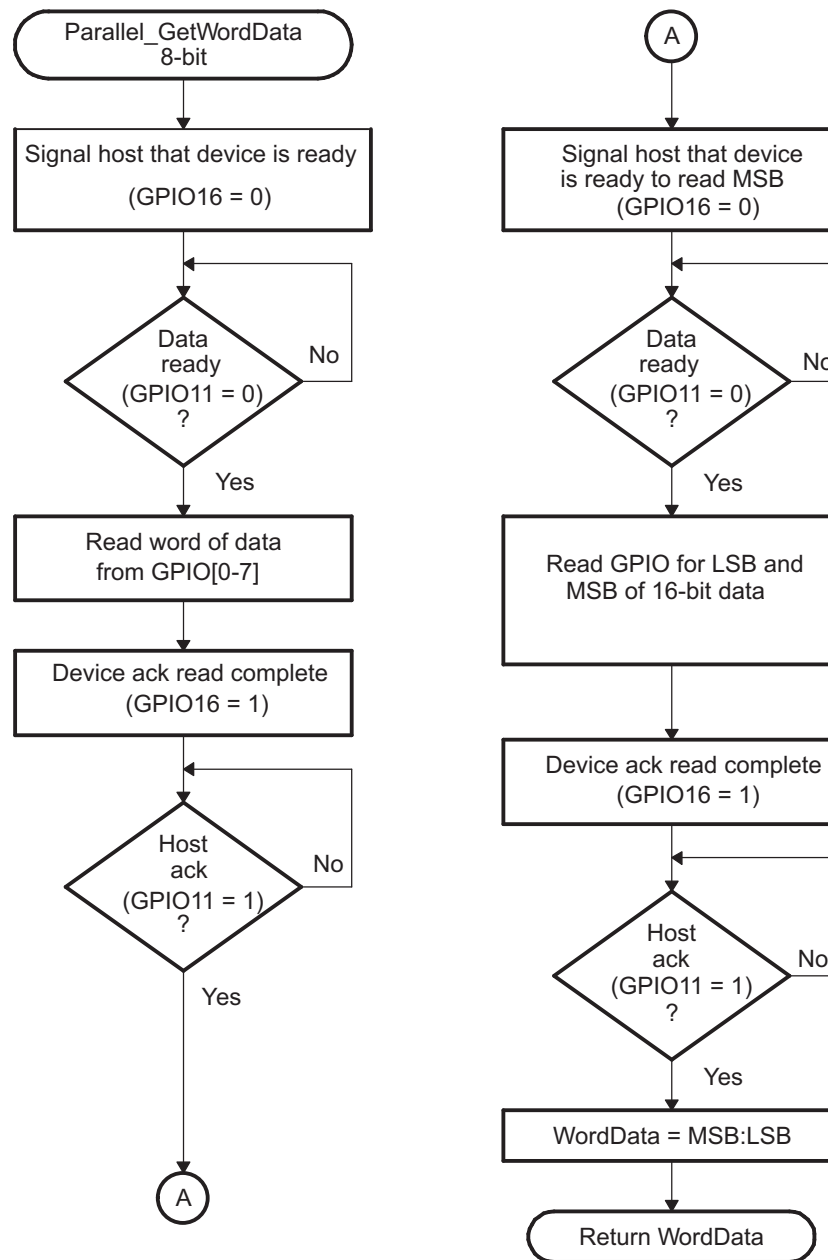


Figure 4-16 shows the flow used to read a single word of data from the parallel port.

- **8-bit data stream**

The 8-bit routine, shown in Figure 4-16, discards the upper 8 bits of the first read from the port and treats the lower 8 bits masked with GPIO7 in bit position 7 and GPIO6 in bit position 6 as the the least significant byte (LSB) of the word to be fetched. The routine will then perform a second read to fetch the most significant byte (MSB). It then combines the MSB and LSB into a single 16-bit value to be passed back to the calling routine.

Figure 4-16. 8-Bit Parallel GetWord Function

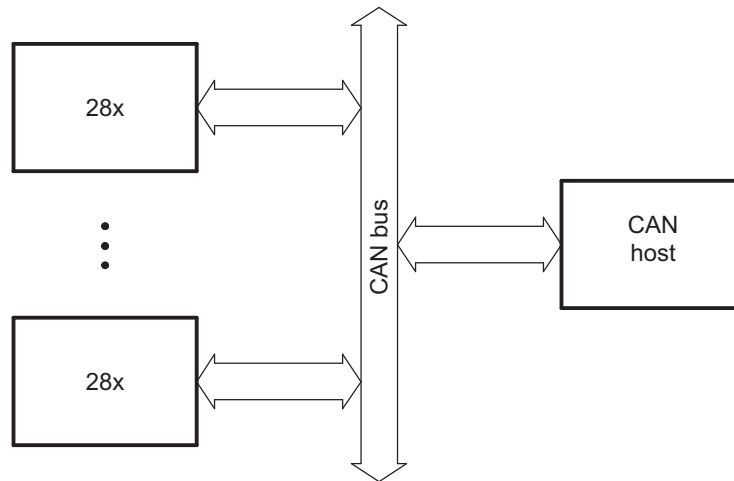


#### 4.6.7.6 CAN Boot Mode

The CAN bootloader asynchronously transfers code from CAN-A to internal memory. The host can be any CAN node. The communication is first done with 11-bit standard identifiers (with a MSGID of 0x1) using two bytes per data frame. The host can download a kernel to reconfigure the CAN if higher data throughput is desired.



Figure 4-17. Overview of CAN-A Bootloader Operation



The bit timing registers are programmed in such a way that a 100 kbps bit rate is achieved with a 20 MHz external oscillator, as shown in [Table 4-25](#).

Table 4-25. Bit-Rate Value for Internal Oscillators

OSCCLK	SYSClk	Bit Rate
20 MHz	10 MHz	100 kbps

The SYSClkOUT values shown are the reset values with the default PLL setting. The BRP and bit-time values are hard-coded to 10 and 20, respectively.

Mailbox 1 is programmed with a standard MSGID of 0x1 for boot-loader communication. The CAN host should transmit only 2 bytes at a time, LSB first and MSB next. For example, to transmit the word 0x08AA to the device, transmit AA first, followed by 08. The program flow of the CAN bootloader is identical to the SCI bootloader. The data sequence for the CAN bootloader is shown in [Table 4-26](#):

Table 4-26. CAN 8-Bit Data Stream

Bytes	Byte 1 of 2	Byte 2 of 2	Description
1 2	AA	08	0x08AA (KeyValue for memory width = 16 bits)
3 4	00	00	reserved
5 6	00	00	reserved
7 8	00	00	reserved
9 10	00	00	reserved
11 12	00	00	reserved
13 14	00	00	reserved
15 16	00	00	reserved
17 18	00	00	reserved
19 20	BB	AA	Entry point PC[22:16]
21 22	DD	CC	Entry point PC[15:0] (PC = 0xAABBCCDD)
23 24	NN	MM	Block size of the first block of data to load = 0xMMNN words
25 26	BB	AA	Destination address of first block Addr[31:16]
27 28	DD	CC	Destination address of first block Addr[15:0] (Addr = 0xAABBCCDD)
29 30	BB	AA	First word of the first block in the source being loaded = 0xAABB
...			....
...			Data for this section.
...			....
.	BB	AA	Last word of the first block of the source being loaded = 0xAABB

**Table 4-26. CAN 8-Bit Data Stream (continued)**

Bytes	Byte 1 of 2	Byte 2 of 2	Description
.	NN	MM	Block size of the 2nd block to load = 0xMMNN words
.	BB	AA	Destination address of second block Addr[31:16]
.	DD	CC	Destination address of second block Addr[15:0]
.	BB	AA	First word of the second block in the source being loaded
.			...
n n+1	BB	AA	Last word of the last block of the source being loaded (More sections if required)
n+2 n+3	00	00	Block size of 0000h - indicates end of the source program

## 4.6.8 Boot Data Stream Structure

This section details the data transfer protocols or stream structures that allow boot data transfer between boot ROM and host device. This data transfer protocol is compatible to the respective bootloaders on the C2000 devices.

### 4.6.8.1 Bootloader Data Stream Structure

The following two tables and associated examples show the structure of the data stream incoming to the bootloader. The basic structure is the same for all the bootloaders and is based on the C54x source data stream generated by the C54x hex utility. The C28x hex utility (hex2000.exe) has been updated to support this structure. The hex2000.exe utility is included with the C2000 code generation tools. All values in the data stream structure are in hex.

The first 16-bit word in the data stream is known as the key value. The key value is used to tell the bootloader the width of the incoming stream: 8 or 16 bits. Note that not all bootloaders will accept both 8 and 16-bit streams. Please refer to the detailed information on each loader for the valid data stream width. For an 8-bit data stream, the key value is 0x08AA and for a 16-bit stream it is 0x10AA. If a bootloader receives an invalid key value, then the load is aborted.

The next eight words are used to initialize register values or otherwise enhance the bootloader by passing values to it. If a bootloader does not use these values then they are reserved for future use and the bootloader simply reads the value and then discards it. Currently only the SPI and I2C and parallel bootloaders use these words to initialize registers.

The tenth and eleventh words comprise the 22-bit entry point address. This address is used to initialize the PC after the boot load is complete. This address is most likely the entry point of the program downloaded by the bootloader.

The twelfth word in the data stream is the size of the first data block to be transferred. The size of the block is defined as 8-bit data stream format. For example, to transfer a block of 20 8-bit data values from an 8-bit data stream, the block size would be 0x000A to indicate 10 16-bit words.

The next two words tell the loader the destination address of the block of data. Following the size and address will be the 16-bit words that makeup that block of data.

This pattern of block size/destination address repeats for each block of data to be transferred. Once all the blocks have been transferred, a block size of 0x0000 signals to the loader that the transfer is complete. At this point the loader will return the entry point address to the calling routine which in turn will cleanup and exit. Execution will then continue at the entry point address as determined by the input data stream contents.

**Table 4-27. LSB/MSB Loading Sequence in 8-Bit Data Stream**

Byte		Contents	
		LSB (First Byte of 2)	MSB (Second Byte of 2)
1	2	LSB: AA (KeyValue for memory width = 8 bits)	MSB: 08h (KeyValue for memory width = 8 bits)
3	4	LSB: Register initialization value or reserved	MSB: Register initialization value or reserved
5	6	LSB: Register initialization value or reserved	MSB: Register initialization value or reserved
7	8	LSB: Register initialization value or reserved	MSB: Register initialization value or reserved
...	...	...	...
...	...	...	...
17	18	LSB: Register initialization value or reserved	MSB: Register initialization value or reserved
19	20	LSB: Upper half of Entry point PC[23:16]	MSB: Upper half of entry point PC[31:24] (Always 0x00)
21	22	LSB: Lower half of Entry point PC[7:0]	MSB: Lower half of Entry point PC[15:8]
23	24	LSB: Block size in words of the first block to load. If the block size is 0, this indicates the end of the source program. Otherwise another block follows. For example, a block size of 0x000A would indicate 10 words or 20 bytes in the block.	MSB: block size
25	26	LSB: MSW destination address, first block Addr[23:16]	MSB: MSW destination address, first block Addr[31:24]
27	28	LSB: LSW destination address, first block Addr[7:0]	MSB: LSW destination address, first block Addr[15:8]
29	30	LSB: First word of the first block being loaded	MSB: First word of the first block being loaded
...	...	...	...
...	...	...	...
.	.	LSB: Last word of the first block to load	MSB: Last word of the first block to load
.	.	LSB: Block size of the second block	MSB: Block size of the second block
.	.	LSB: MSW destination address, second block Addr[23:16]	MSB: MSW destination address, second block Addr[31:24]
.	.	LSB: LSW destination address, second block Addr[7:0]	MSB: LSW destination address, second block Addr[15:8]
.	.	LSB: First word of the second block being loaded	MSB: First word of the second block being loaded
...	...	...	...
...	...	...	...
.	.	LSB: Last word of the second block	MSB: Last word of the second block
.	.	LSB: Block size of the last block	MSB: Block size of the last block
.	.	LSB: MSW of destination address of last block Addr[23:16]	MSB: MSW destination address, last block Addr[31:24]
.	.	LSB: LSW destination address, last block Addr[7:0]	MSB: LSW destination address, last block Addr[15:8]
.	.	LSB: First word of the last block being loaded	MSB: First word of the last block being loaded
...	...	...	...
...	...	...	...
.	.	LSB: Last word of the last block	MSB: Last word of the last block
n	n+1	LSB: 00h	MSB: 00h - indicates the end of the source

**Example 4-1. Data Stream Structure 8-bit**

```

AA 08          ; 0x08AA 8-bit key value
00 00 00 00   ; 8 reserved words
00 00 00 00
00 00 00 00
00 00 00 00
3F 00 00 80   ; 0x003F8000 EntryAddr, starting point after boot load completes
05 00         ; 0x0005 - First block consists of 5 16-bit words
3F 00 10 90   ; 0x003F9010 - First block will be loaded starting at 0x3F9010
01 00        ; Data loaded = 0x0001 0x0002 0x0003 0x0004 0x0005
02 00
03 00
04 00
05 00
02 00        ; 0x0002 - 2nd block consists of 2 16-bit words

```

**Example 4-1. Data Stream Structure 8-bit (continued)**

```

3F 00 00 80 ; 0x003F8000 - 2nd block will be loaded starting at 0x3F8000
00 77      ; Data loaded = 0x7700 0x7625
25 76
00 00      ; 0x0000 - Size of 0 indicates end of data stream
    
```

After load has completed the following memory values will have been initialized as follows:

Location	Value
0x3F9010	0x0001
0x3F9011	0x0002
0x3F9012	0x0003
0x3F9013	0x0004
0x3F9014	0x0005
0x3F8000	0x7700
0x3F8001	0x7625

PC Begins execution at 0x3F8000

### 4.6.9 GPIO Assignments

This section details the GPIOs and boot options used for each boot mode set in BOOT\_DEFx located at Z1-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH. Refer to [Configuring Alternate Boot Mode Pins](#) on how to manipulate BOOT\_DEFx. When selecting a boot mode option, make sure to verify that the necessary pins are available in the pin mux options for the specific device package being used.

**Table 4-28. SCI Boot Options**

Option	BOOTDEFx Value	SCIATX GPIO	SCIARX GPIO
0 (default)	0x01	GPIO29	GPIO28
1	0x21	GPIO16	GPIO17
2	0x41	GPIO8	GPIO9
3	0x61	GPIO48	GPIO49
4	0x81	GPIO24	GPIO25

**NOTE:** Pull-ups are enabled on the SCIATX and SCIARX pins.

**Table 4-29. CAN Boot Options**

Option	BOOTDEFx Value	CANTXA GPIO	CANRXA GPIO
0 (default)	0x02	GPIO32	GPIO33
1	0x22	GPIO4	GPIO5
2	0x42	GPIO31	GPIO30
3	0x62	GPIO37	GPIO35

**NOTE:** Pull-ups are enabled on the CANTXA and SCIARX pins.

**Table 4-30. Flash Boot Options**

Option	BOOTDEFx Value	Flash Entry Point (Address)	Flash Bank, Sector
0 (default)	0x03	Flash – Default Option 1 (0x00080000)	Bank 0, Sector 0
1	0x23	Flash – Option 2 (0x0008E000)	Bank 0, Sector 14
2	0x43	Flash – Option 3 (0x00090000)	Bank 1, Sector 0

**Table 4-30. Flash Boot Options (continued)**

Option	BOOTDEFx Value	Flash Entry Point (Address)	Flash Bank, Sector
3	0x63	Flash – Option 4 (0x0009EFF0)	Bank 1, Sector 14

**Table 4-31. Wait Boot Options**

Option	BOOTDEFx Value	Watchdog Status
0	0x04	Enabled
1	0x24	Disabled

**Table 4-32. SPI Boot Options**

Option	BOOTDEFx Value	SPIA_SIMO	SPIA_SOMI	SPIA_CLK	SPIA_STE
1	0x26	GPIO8	GPIO10	GPIO9	GPIO11
2	0x46	GPIO54	GPIO55	GPIO56	GPIO57
3	0x66	GPIO16	GPIO17	GPIO56	GPIO57
4	0x86	GPIO8	GPIO17	GPIO9	GPIO11

---

**NOTE:** Pull-ups are enabled on the SPIA\_SIMO, SPIA\_SOMI, SPIA\_CLK, and SPIA\_STE pins.

---

**Table 4-33. I2C Boot Options**

Option	BOOTDEFx Value	SDAA GPIO	SCLA GPIO
0	0x07	GPIO32	GPIO33
1	0x47	GPIO26	GPIO27
2	0x67	GPIO42	GPIO43

---

**NOTE:** Pull-ups are enabled on the SDAA and SCLA pins.

---

**Table 4-34. Parallel Boot Options**

Option	BOOTDEFx Value	D0-D7 GPIO	DSP Control GPIO	Host Control GPIO
0 (default)	0x00	GPIO0-GPIO7	GPIO16	GPIO11

---

**NOTE:** Pull-ups are enabled on GPIO0 to GPIO7.

---

**Table 4-35. RAM Boot Options**

Option	BOOTDEFx Value	RAM Entry Point (Address)
0	0x05	0x00000000

#### 4.6.10 DCSM Usage

This section explains how the bit field values from the user-configurable DCSM OTP location, Z1-OTP-BOOT-GPREG2, are decoded by the boot ROM after DCSM initialization is complete.

**Table 4-36. DCSM Z1-OTP-BOOT-GPREG2 Bit Fields**

Bit	Name	Description	Boot ROM Action
31-24	Key	Write 0x5A to these 8-bits to tell the boot ROM code that the bits in this register are valid.	If user set to 0x5A, boot ROM uses the values in this register. If set to any other value, boot ROM ignores the values in this register.
23-8	Reserved	Reserved	No action
7-6	Reserved	Reserved	No action
5-4	Error Status Pin	Sets the GPIO pin to be used as the ERRORSTS 0x0 – GPIO24 0x1 – GPIO28 0x2 – GPIO29 0x3 – ERRORSTS disabled (Default)	Boot ROM configures the appropriate mux for the selected GPIO pin.
3-0	Reserved		

**NOTE:** Users should program user DCSM OTP locations Z1-OTP-BOOT-GPREG2 and Z1-OTP-BOOTPIN-CONFIG at the same time, because they share ECC.

#### 4.6.11 Clock Initialization

During boot up, the boot ROM initializes the device clocking, depending upon the reset source, to assist in faster boot time response. Clock configurations are performed by the boot ROM code only for POR or XRS reset types. For all other resets, the boot ROM starts executing with the clocks that were already set up before reset.

**Table 4-37. Boot Clock Sources**

Source	Frequency	Description
INTOSC2	10 MHz	Default clock source
INTOSC1	10 MHz	Set as clock source if missing clock is detected at power up or right after device reset

**Table 4-38. Clock State after Boot ROM**

Reset Source	Clock State
POR/XRS	Bypassed PLL. PLL multiplier is set to 0x0. Clock divider is set to /1.
All other Resets	Maintain clocks setup before device reset.

**NOTE:** When the PLL is used by the bootloader, it is bypassed by the boot ROM code before branching to the user application.

#### 4.6.12 Boot Status Information

Boot ROM keeps a record of the different events that can occur during boot ROM execution. This is because NMI and other exceptions are enabled by default in the device, and must be handled accordingly. Boot ROM stores the boot status information in a RAM location so that the user application can look at this boot status and take the necessary actions per the application's needs to handle these events.

#### 4.6.12.1 Booting Status

This section details the boot status RAM location and its bit field definitions. When the specific bit field is set, the described event or action has occurred.

**Table 4-39. Boot Status Address**

Description	Address
BROM_STATUS	0x0000 0002

**Table 4-40. Boot Status Bit Fields**

Bit	Description
22	Boot ROM detected a missing clock NMI
21	Boot ROM detected a RAM bit error NMI
20	Boot ROM detected a Flash bit error NMI
17	Boot ROM detected a PIE mismatch
16	Boot ROM detected an ITRAP
15	Boot ROM has completed running
13	Boot ROM handled POR
12	Boot ROM handled XRS
11	Boot ROM handled all the resets
10	POR memory test has completed
9	DCSM initialization has completed
7	CAN boot has started
6	I2C boot has started
5	SPI boot has started
4	SCI boot has started
3	RAM boot has started
2	Parallel boot has started
1	Flash boot has started
0	Boot ROM has started running

#### 4.6.12.2 Flash Single Bit Error Status

After DCSM is initialized during a reset by the boot ROM, the flash single bit error status is stored at the high and low RAM location detailed in this section. Both the high and low status are 32 bits wide.

**Table 4-41. Flash Single Bit Error Status Addresses**

Description	Address
Flash Single Bit Error (Low)	0x0000 0004
Flash Single Bit Error (High)	0x0000 0006

#### 4.6.12.3 PBIST Status

When PBIST is enabled to run, the status of the test is set within a status variable in boot ROM. This variable is reset on every run of boot ROM. If the user application requires the preservation of this status, the PBIST status must be copied to the application's RAM

#### 4.6.13 ROM Version

The ROM revision and release date information is stored at the ROM locations specified in this section.

**Table 4-42. Boot ROM Version Information**

Start Address	End Address	Contents
0x003F FF7A	0x003F FF7A	Revision Number
0x003F FF7B	0x003F FF7B	Revision Date

Interpreting the contents:

- Reading a revision number value of “0x100” represents version “1.0”.
- Reading a revision date value of “0x0715” represents “07/15” or “July 2015”.

## 4.7 Application Notes for Using the Bootloaders

### 4.7.1 The C2000 Hex Utility

To use the features of the bootloader, you must generate a data stream and boot table as described in [Bootloader Data Stream Structure](#). The hex conversion utility tool, included with the 28x code generation tools, can generate the required data stream including the required boot table. This section describes the hex2000 utility. An example of a file conversion performed by hex2000 is described in [Example 4-2](#).

The hex utility supports creation of the boot table required for the SCI, SPI, I2C, CAN, and parallel I/O loaders. That is, the hex utility adds the required information to the file such as the key value, reserved bits, entry point, address, block start address, block length and terminating value. The contents of the boot table vary slightly depending on the boot mode and the options selected when running the hex conversion utility. The actual file format required by the host (ASCII, binary, hex, etc.) will differ from one specific application to another and some additional conversion may be required.

To build the boot table, follow these steps:

#### 1. Assemble or compile the code.

This creates the object files that will then be used by the linker to create a single output file.

#### 2. Link the file.

The linker combines all of the object files into a single output file in common object file format (COFF). The specified linker command file is used by the linker to allocate the code sections to different memory blocks. Each block of the boot table data corresponds to an initialized section in the COFF file. Uninitialized sections are not converted by the hex conversion utility. The following options may be useful:

The linker `-m` option can be used to generate a map file. This map file will show all of the sections that were created, their location in memory and their length. It can be useful to check this file to make sure that the initialized sections are where you expect them to be.

The linker `-w` option configures the linker to show if the linker assigned a section to a memory region automatically. For example, if you have a section in your code called `ramfuncs`.

#### 3. Run the hex conversion utility.

Choose the appropriate options for the desired boot mode and run the hex conversion utility to convert the COFF file produced by the linker to a boot table.

See the *TMS320C28x Assembly Language Tools User's Guide* ([SPRU513](#)) and the *TMS320C28x Optimizing C/C++ Compiler User's Guide* ([SPRU514](#)) for more information on the compiling and linking process.

[Table 4-43](#) summarizes the hex conversion utility options available for the bootloader. See the *TMS320C28x Assembly Language Tools User's Guide* ([SPRU513](#)) for a detailed description of the hex2000 operations used to generate a boot table. Updates will be made to support the I2C boot. See the Codegen release notes for the latest information.

**Table 4-43. Boot Loader Options**

Option	Description
<code>-boot</code>	Convert all sections into bootable form (use instead of a SECTIONS directive)
<code>-sci8</code>	Specify the source of the bootloader table as the SCI-A port, 8-bit mode



**Table 4-43. Boot Loader Options (continued)**

Option	Description
-spi8	Specify the source of the bootloader table as the SPI-A port, 8-bit mode
-gpio8	Specify the source of the bootloader table as the GPIO port, 8-bit mode
-bootorg value	Specify the source address of the bootloader table
-lospcp value	Specify the initial value for the LOSPCP register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F.
-spibrr value	Specify the initial value for the SPIBRR register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F.
-e value	Specify the entry point at which to begin execution after boot loading. The value can be an address or a global symbol. This value is optional. The entry point can be defined at compile time using the linker -e option to assign the entry point to a global symbol. The entry point for a C program is normally <code>_c_int00</code> unless defined otherwise by the -e linker option.
-i2c8	Specify the source of the bootloader table as the I2C-A port, 8-bit
-i2cpsc value	Specify the value for the I2CPSC register. This value will be loaded and take effect after all I2C options are loaded, prior to reading data from the EEPROM. This value will be truncated to the least significant eight bits and should be set to maintain an I2C module clock of 7-12 MHz.
-i2cclkh value	Specify the value for the I2CCLKH register. This value will be loaded and take effect after all I2C options are loaded, prior to reading data from the EEPROM.
-i2cckl value	Specify the value for the I2CCLKL register. This value will be loaded and take effect after all I2C options are loaded, prior to reading data from the EEPROM.

**Example 4-2. HEX2000.exe Command Syntax**

```
C: HEX2000 GPIO34TOG.OUT -boot -gpio8 -a
```

Where:

- boot Convert all sections into bootable form.
- gpio8 Use the GPIO in 8-bit mode data format. The eCAN uses the same data format as the GPIO in 8-bit mode.
- a Select ASCII-Hex as the output format.

## Control Law Accelerator (CLA)

---



---

The Control Law Accelerator (CLA) Type-2 is an independent, fully-programmable, 32-bit floating-point math processor that brings concurrent control-loop execution to the C28x family. The low interrupt latency of the CLA allows it to read ADC samples "just-in-time." This significantly reduces the ADC sample to output delay to enable faster system response and higher MHz control loops. By using the CLA to service time-critical control loops, the main CPU is free to perform other system tasks such as communications and diagnostics. This chapter provides an overview of the architectural structure and components of the control law accelerator.

Topic	Page
<b>5.1 Introduction .....</b>	<b>643</b>
<b>5.2 Features .....</b>	<b>643</b>
<b>5.3 CLA Interface .....</b>	<b>645</b>
<b>5.4 CLA, DMA, and CPU Arbitration .....</b>	<b>650</b>
<b>5.5 CLA Configuration and Debug .....</b>	<b>652</b>
<b>5.6 Pipeline .....</b>	<b>657</b>
<b>5.7 Instruction Set.....</b>	<b>663</b>
<b>5.8 CLA Registers .....</b>	<b>781</b>

## 5.1 Introduction

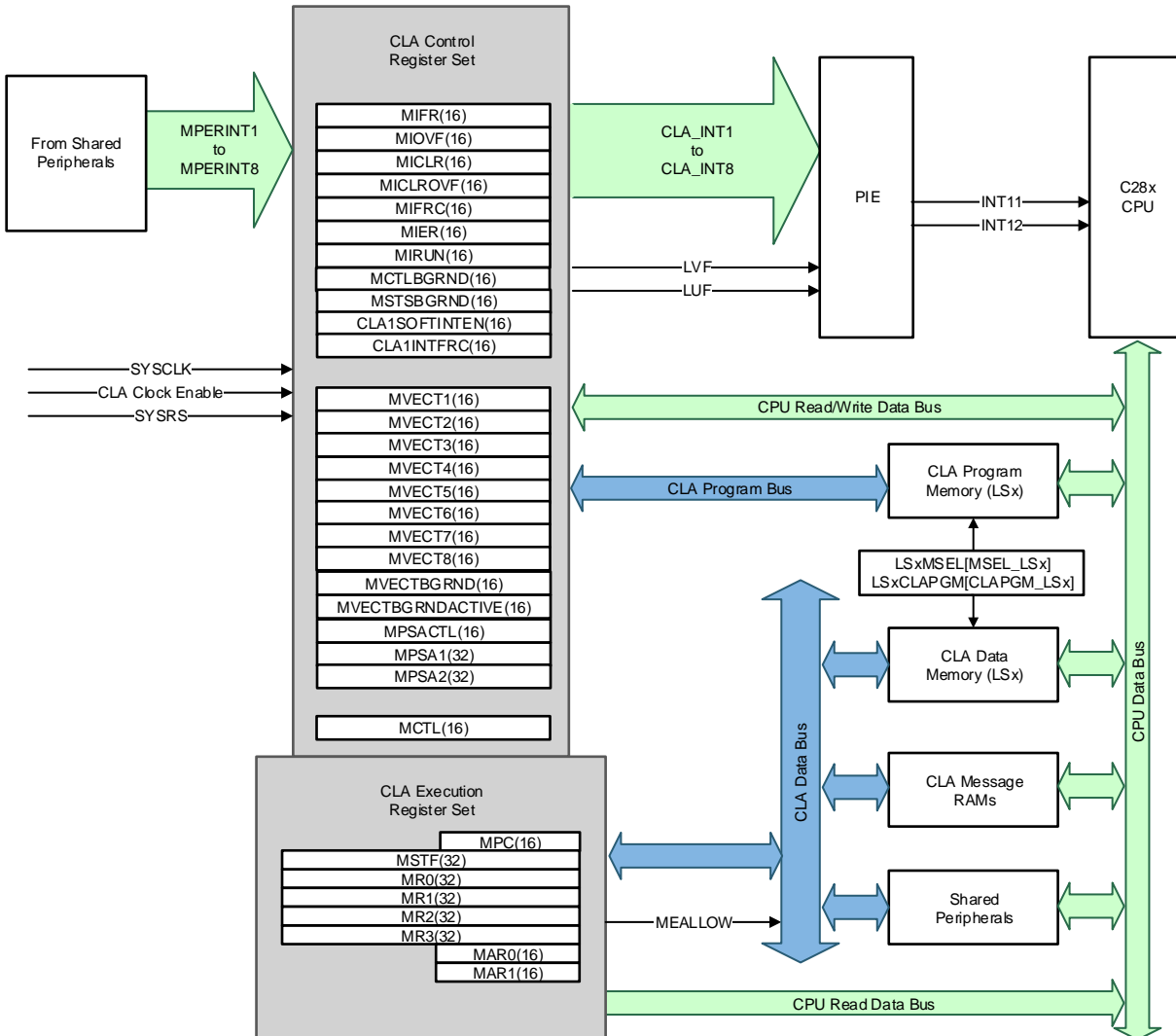
The Control Law Accelerator extends the capabilities of the C28x CPU by adding parallel processing. Time-critical control loops serviced by the CLA can achieve low ADC sample to output delay. Thus, the CLA enables faster system response and higher frequency control loops. Utilizing the CLA for time-critical tasks frees up the main CPU to perform other system and communication functions concurrently.

## 5.2 Features

The following is a list of major features of the CLA:

- C compilers are available for CLA software development.
- Clocked at the same rate as the main CPU (SYSCLKOUT).
- An independent architecture allowing CLA algorithm execution independent of the main C28x CPU.
  - Complete bus architecture:
    - Program Address Bus (PAB) and Program Data Bus (PDB)
    - Data Read Address Bus (DRAB), Data Read Data Bus (DRDB), Data Write Address Bus (DWAB), and Data Write Data Bus (DWDB)
  - Independent eight stage pipeline.
  - 16-bit program counter (MPC)
  - Four 32-bit result registers (MR0-MR3)
  - Two 16-bit auxiliary registers (MAR0, MAR1)
  - Status register (MSTF)
- Instruction set includes:
  - IEEE single-precision (32-bit) floating point math operations
  - Floating-point math with parallel load or store
  - Floating-point multiply with parallel add or subtract
  - 1/X and 1/sqrt(X) estimations
  - Data type conversions.
  - Conditional branch and call
  - Data load/store operations
- The CLA program code can consist of up to eight tasks or interrupt service routines, or 7 tasks and a main background task.
  - The start address of each task is specified by the MVECT registers.
  - No limit on task size as long as the tasks fit within the configurable CLA program memory space.
  - One task is serviced at a time until its completion. There is no nesting of tasks.
  - Upon task completion a task-specific interrupt is flagged within the PIE.
  - When a task finishes the next highest-priority pending task is automatically started.
  - The Type-2 CLA can have a main task that runs continuously in the background, while other high priority events trigger a foreground task.
- Task trigger mechanisms:
  - C28x CPU via the IACK instruction
  - Task1 to Task8: up to 256 possible trigger sources from peripherals connected to the shared bus on which the CLA assumes secondary ownership.
  - Task8 can be set to be the background task, while Tasks 1 through 7 take peripheral triggers.
- Memory and Shared Peripherals:
  - Two dedicated message RAMs for communication between the CLA and the main CPU.
  - Two dedicated message RAMs for communication between the CLA and the DMA.
  - The C28x CPU can map CLA program and data memory to the main CPU space or CLA space.

Figure 5-1. CLA (Type 2) Block Diagram



## 5.3 CLA Interface

This chapter describes how the C28x main CPU can interface to the CLA and vice versa.

### 5.3.1 CLA Memory

The CLA can access three types of memory: program, data and message RAMs. The behavior and arbitration for each type of memory is described in detail below. The CLA RAMs are protected by the DCSM module. Refer to the *DCSM* section of this manual for more details on the security scheme.

- **CLA Program Memory**

The CLA program can be loaded any of the local shared memories (LSxRAM). At reset, all memory blocks are mapped to the CPU. While mapped to the CPU space, the CPU can copy the CLA program code into the memory. During debug, the memory can also be loaded directly by Code Composer Studio™.

Once the memory is initialized with CLA code, the CPU maps it to the CLA program space by:

1. Assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL\_LSx] bit.
2. Specifying the memory block as a code block for the CLA by writing a 1 to the MemCfgRegs.LSxCLAPGM[CLAPGM\_LSx] bit.

When a memory block is configured as CLA program memory, debug accesses are allowed only on cycles where the CLA is not fetching a new instruction. A detailed explanation of the memory configurations and access arbitration (CPU, CLA, and DEBUG) process can be found in [Section 3.11](#).

All CLA program fetches are performed as 32-bit read operations and all opcodes must be aligned to an even address. Since all CLA opcodes are 32-bits, this alignment occurs naturally.

- **CLA Data Memory**

Any of the device's LSxRAMs can serve as data memory blocks to the CLA. At reset, all blocks are mapped to the CPU memory space, whereby the CPU can initialize the memory with data tables, coefficients, and so on, for the CLA to use.

Once the memory is initialized with CLA data, the CPU maps it to the CLA data space by:

1. Assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL\_LSx] bit.
2. Specifying the memory block as a data block for the CLA by writing a 0 to the MemCfgRegs.LSxCLAPGM[CLAPGM\_LSx] bit. The value of this bit at reset is 0.

When a memory block is configured as CLA data memory, CLA read and write accesses are arbitrated along with CPU accesses. The user has the option of turning on CPU fetch or write protection to the memory by writing to the appropriate bits of the MemCfgRegs.LSxACCPROT<sub>x</sub> registers. A detailed explanation of the memory configurations and access arbitration (CPU, CLA, and DEBUG) process can be found in [Section 3.11](#).

- **CLA Shared Message RAMs**

There are two memory blocks for data sharing and communication between the CLA and the CPU. The message RAMs are always mapped to both CPU and CLA memory spaces, and only data access is allowed; no program fetches can be performed.

- **CLA to CPU Message RAM**

The CLA can use this block to pass data to the CPU. This block is both readable and writable by the CLA. This block is also readable by the CPU but writes by the CPU are ignored.

- **CPU to CLA Message RAM**

The CPU can use this block to pass data and messages to the CLA. This message RAM is both readable and writable by the CPU. The CLA can perform reads but writes by the CLA are ignored.

### 5.3.2 CLA Memory Bus

The CLA has dedicated bus architecture similar to that of the C28x CPU where there are separate program read, data read, and data write buses. Thus, there can be simultaneous instruction fetch, data read, and data write in a single cycle. Like the C28x CPU, the CLA expects memory logic to align any 32-bit read or write to an even address. If the address-generation logic generates an odd address, the CLA will begin reading or writing at the previous even address. This alignment does not affect the address values generated by the address-generation logic.

- **CLA Program Bus**

The CLA program bus has an access range of 32K 32-bit instructions. Since all CLA instructions are 32 bits, this bus always fetches 32 bits at a time and the opcodes must be even-word aligned. The amount of program space available for the CLA is limited to the number of available LSxRAM blocks. This number is device-dependent and will be described in the device-specific data manual.

- **CLA Data Read Bus**

The CLA data read bus has a 64K x 16 address range. The bus can perform 16 or 32-bit reads and will automatically stall if there are memory access conflicts. The data read bus has access to both the message RAMs, CLA data memory, and the shared peripherals.

- **CLA Data Write Bus**

The CLA data write bus has a 64K x 16 address range. This bus can perform 16 or 32-bit writes. The bus will automatically stall if there are memory access conflicts. The data write bus has access to the CLA to CPU message RAM, CLA data memory, and the shared peripherals.

### 5.3.3 Shared Peripherals and EALLOW Protection

For a given CPU subsystem, the CPU, CLA, and DMA can share access to some peripherals. There is a 3-way arbitration among the different master's that is described in [Section 5.4](#). Each peripheral has an access control register with two bit fields, CPU<sub>n</sub>AC, CLAnAC, and DMA<sub>n</sub>AC (n being the instance) that determine what kind of access is given to that particular master.

**Note:** The CLA read access time to the bus is 2-wait states while write access is 0-wait.

Refer to the device data manual for the list of peripherals connected to the bus.

Several peripheral control registers are protected from spurious 28x CPU writes by the EALLOW protection mechanism. These same registers are also protected from spurious CLA writes. The EALLOW bit in the CPU status register 1 (ST1) indicates the state of protection for the CPU. Likewise the MEALLOW bit in the CLA status register (MSTF) indicates the state of write protection for the CLA. The MEALLOW CLA instruction enables write access by the CLA to EALLOW protected registers. Likewise the MEDIS CLA instruction will disable write access. This way the CLA can enable/disable write access independent of the CPU.

The ADC offers the option to generate an early interrupt pulse at the start of a sample conversion. If this option is used to start an ADC-triggered CLA task, the user may use the intervening cycles, until the completion of the conversion, to perform preliminary calculations or loads and stores before finally reading the ADC value. The CLA pipeline activity for this scenario is shown in [Section 5.6](#).

### 5.3.4 CLA Tasks and Interrupt Vectors

The CLA program code is divided up into tasks or interrupt service routines. Tasks do not have a fixed starting location or length. The CLA program memory can be divided up as desired. The CLA uses the contents of the interrupt vectors (MVECT1 to MVECT8) to determine where a task begins; tasks are terminated by the MSTOP instruction.

The CLA supports eight tasks. Task 1 has the highest priority and task 8 has the lowest priority. The Type-2 CLA offers the option of setting the lowest priority task, for example, Task 8, as a background task that, once triggered, runs continuously until the user either terminates it or resets the CLA or the device. The remaining tasks, 1 through 7, maintain their priority levels and interrupt the background task when triggered.

The background task is enabled by setting the BGEN bit in the MCTLBGRND register; this causes the hardware to disable task 8 in the MIER register. The background task derives its interrupt vector from the MVECTBGRND register instead of MVECT8.

A task can be requested by a peripheral interrupt or by software:

- **Peripheral interrupt trigger**

Each task can be triggered by software-selectable interrupt sources. The trigger for each task is defined by writing an appropriate value to the `DmaClaSrcSelRegs.CLA1TASKSRCSELx[TASKx]` bit field. Each option specifies an interrupt source from a specific peripheral on the shared bus. The peripheral interrupt triggers are listed in [Table 5-1](#).

**Table 5-1. Configuration Options**

Select Value (8-bit)	CLA Trigger Source
0	Software Trigger
1	ADCA1_INT
2	ADCA2_INT
3	ADCA3_INT
4	ADCA4_INT
5	ADCA_EVT_INT
6	ADCB1_INT
7	ADCB2_INT
8	ADCB3_INT
9	ADCB4_INT
10	ADCB_EVT_INT
11	ADCC1_INT
12	ADCC2_INT
13	ADCC3_INT
14	ADCC4_INT
15	ADCC_EVT_INT
28:16	Reserved
29	XINT1
30	XINT2
31	XINT3
32	XINT4
33	XINT5
35:34	Reserved
36	EPWM1INT
37	EPWM2INT
38	EPWM3INT
39	EPWM4INT
40	EPWM5INT
41	EPWM6INT

**Table 5-1. Configuration Options (continued)**

Select Value (8-bit)	CLA Trigger Source
42	EPWM7INT
43	EPWM8INT
67:44	Reserved
68	TINT0
69	TINT1
70	TINT2
71	Reserved
72	Reserved
73	Reserved
74	Reserved
75	ECAP1INT
76	ECAP2INT
77	ECAP3INT
78	ECAP4INT
79	ECAP5INT
80	ECAP6INT
81	ECAP7INT
82	Reserved
83	EQEP1INT
84	EQEP2INT
91:85	Reserved
92	ECAP6INT2
93	ECAP7INT2
94	Reserved
95	SD1INT
96	SD1DRINT1
97	SD1DRINT2
98	SD1DRINT3
99	SD1DRINT4
104:100	Reserved
105	PMBUSINT
108:106	Reserved
109	SPITXINTA
110	SPIRXINTA
111	SPITXINTB
112	SPIRXINTB
116:113	Reserved
117	LINA_INT1
118	LINA_INT0
120:119	Reserved
121	CLA1PROMCRC
122	Reserved
123	FSITX_INT1
124	FSITX_INT2
125	FSIRX_INT1
126	FSIRX_INT2
127	CLB1INT



**Table 5-1. Configuration Options (continued)**

Select Value (8-bit)	CLA Trigger Source
128	CLB2INT
129	CLB3INT
130	CLB4INT
255:131	Reserved

For example, Task 1 (MVECT1) can be set to trigger on EPWMINT1 by writing 36 to DmaClaSrcSelRegs.CLA1TASKSRCSEL1.bit.TASK1. To disable the triggering of a task by a peripheral, the user must set the DmaClaSrcSelRegs.CLA1TASKSRCSELx[TASKx] bit field to 0. It should be noted that a CLA task only triggers on a level transition (an edge) of the configured interrupt source.

- **Software Trigger**

CPU software can trigger tasks by writing to the MIFRC register or by the IACK instruction. Using the IACK instruction is more efficient because it does not require you to issue an EALLOW to set MIFR bits. Set the MCTL[IACKE] bit to enable the IACK feature. Each bit in the operand of the IACK instruction corresponds to a task. For example IACK #0x0001 will set bit 0 in the MIFR register to start task 1. Likewise IACK #0x0003 will set bits 0 and 1 in the MIFR register to start task 1 and task 2.

- **Background Task**

The Type-2 CLA allows the user to use Task 8 as a background task that runs continuously until the user disables it or resets the device (or the CLA using a soft reset). The background task vector is given by the MVECTBGRND register and its operation controlled by the MCTLBGRND register; it is enabled by setting the BGEN bit to 1. The user can then kick off the task through software by writing a 1 to the BGSTART bit (TRIGEN must be 0), or through a peripheral by setting the TRIGEN bit to 1 and then setting the trigger source in the bit-field, DmaClaSrcSelRegs.CLA1TASKSRCSEL2.bit.TASK8. By default the background task is interruptible; the highest priority pending task will be executed first. When a task completes, and there aren't any pending tasks, the execution returns to the background task. The CLA keeps track of the branching point by saving the return address to the MVECTBGRNDACTIVE register, and then popping this address to the MPC when execution returns. The user may choose to make sections of the background task un-interruptible; it is possible to do this with the MSETC BGINTM assembly instruction.

Subsequently, enabling interrupts with the MCLRC BGINTM instruction.

The background interrupt mask bit, BGINTM, can be queried in the MSTSBGRND register. This register also provides the current status of the background task; if it is currently executing, the RUN bit is set to 1, if another trigger for the background task is received while it has already started the overflow (BGOVF) bit is set.

The CLA has its own fetch mechanism and can run and execute a task independently of the CPU. Only one task is serviced at a time; there is no nesting of tasks unless the background task is enabled, then one level of nesting is possible. The task currently running is indicated in the MIRUN register; if the background task is enabled, and running, it is reflected in the MSTSBGRND register (the RUN bit).

Interrupts that have been received but not yet serviced are indicated in the flag register (MIFR). If an interrupt request from a peripheral is received and that same task is already flagged, then the overflow flag bit is set. Overflow flags will remain set until they are cleared by the CPU. If the CLA is idle (no task is currently running) or is executing the background task, then the highest priority interrupt request that is both flagged (MIFR) and enabled (MIER) will start. The flow is as follows:

1. The associated RUN register bit is set (MIRUN) and the flag bit (MIFR) is cleared.
2. The CLA begins execution at the location indicated by the associated interrupt vector (MVECTx). MVECT contains the absolute 16-bit address of the task in the lower 64K memory space. If a task is interrupting the background task then the current program address is stored in the MVECTBGRNDACTIVE register before execution jumps to the task; this saved address is restored to the MPC when the task completes and execution returns to the background task.
3. The CLA executes instructions until the MSTOP instruction is found. This indicates the end of the task.
4. The MIRUN bit is cleared.
5. The task-specific interrupt to the PIE is issued. This informs the main CPU that the task has

completed.

6. The CLA returns to idle (or to the background task, if enabled). Once a task completes the next highest-priority pending task is automatically serviced and this sequence repeats.

### 5.3.5 CLA Software Interrupt to CPU

The CLA can issue a software interrupt to the C28x CPU at any point in the code through the use of the CLA1SOFTINTEN and CLA1INTFRC registers. Please see [Section 5.8](#) for a description of these registers. If a software interrupt is selected for a CLA task, then an end-of-task interrupt will not be issued to the C28x when that task completes.

## 5.4 CLA, DMA, and CPU Arbitration

Typically, CLA activity is independent of the CPU activity. Under the circumstance where the CLA, DMA, or CPU attempt to concurrently access memory or a peripheral register within the same interface, an arbitration procedure will occur. This section describes this arbitration.

The arbitration follows a fixed arbitration scheme with highest priority first:

1. DMA WRITE
2. DMA READ
3. CLA WRITE
4. CLA READ
5. CPU WRITE
6. CPU READ

They are covered in detail in the [Section 3.11](#).

### 5.4.1 CLA Message RAM

Message RAMs consist of four blocks:

- CLA to CPU Message RAM
- CPU to CLA Message RAM
- DMA to CLA Message RAM
- CLA to DMA Message RAM

These blocks are useful for passing data between the CLA and CPU. No opcode fetches, from either the CLA or CPU, are allowed from the message RAMs. A write protection violation will not be generated if the CLA attempts to write to the CPU to CLA or DMA to CLA message RAM but the write will be ignored. The arbitration scheme for the message RAMs are the same as those for the shared memories described in the [Section 3.11](#).

The message RAMs have the following characteristics:

- CLA to CPU Message RAM:  
The following accesses are allowed:
  - CPU reads
  - CLA data reads and writes
  - CPU debug reads and writes

The following accesses are ignored:

  - CPU writes
- CPU to CLA Message RAM:  
The following accesses are allowed:
  - CPU reads and writes
  - CLA reads
  - CPU debug reads and writes

The following accesses are ignored:

- CLA writes

## 5.5 CLA Configuration and Debug

This section discusses the steps necessary to configure and debug the CLA.

### 5.5.1 Building a CLA Application

The control law accelerator can be programmed in either CLA assembly code, using the instructions described in [Section 5.7](#), or a reduced subset of the C language. CLA assembly code resides in the same project with C28x code. The only restriction is the CLA code must be in its own assembly section. This can be easily done using the `.sect` assembly directive. This does not prevent CLA and C28x code from being linked into the same memory region in the linker command file.

System and CLA initialization are performed by the main CPU. This would typically be done in C or C++ but can also include C28x assembly code. The main CPU will also copy the CLA code to the program memory and, if needed, initialize the CLA data RAM(s). Once system initialization is complete and the application begins, the CLA will service its interrupts using the CLA assembly code (or tasks). The main CPU can perform other tasks concurrently with CLA program execution.

The CLA Type 2 requires Codegen VXX.X.X or later with the compiler switch: `--cla_support=cla2`.

### 5.5.2 Typical CLA Initialization Sequence

A typical CLA initialization sequence is performed by the main CPU as described in this section.

#### 1. Copy CLA code into the CLA program RAM

The source for the CLA code can initially reside in the flash or a data stream from a communications peripheral or anywhere the main CPU can access it. The debugger can also be used to load code directly to the CLA program RAM during development.

#### 2. Initialize CLA data RAM, if necessary

Populate the CLA data RAM with any required data coefficients or constants.

#### 3. Configure the CLA registers

Configure the CLA registers, but keep interrupts disabled until later (leave MIER = 0):

- **Enable the CLA peripheral clock using the assigned PCLKCRn register**

The peripheral clock control (PCLKCRn) registers are defined in the *System Control* chapter.

- **Populate the CLA task interrupt vectors**

- MVECT1 to MVECT8

Each vector needs to be initialized with the start address of the task to be executed when the CLA receives the associated interrupt. This address is the full 16-bit starting address of the task in the lower 64K section of memory.

- MVECT1 to MVECT7, and MVECTBGRND

When using the background task, its vector (MVECTBGRND) must be loaded with the start address of the task in lower 64K of memory. Note that Task 8 is ignored when the background task is enabled

- **Select the task interrupt sources**

For each task select the interrupt source in the CLA1TASKSRCSELx register. If a task is software triggered, select no interrupt. Since the background task takes the place of Task 8, it will use the same peripheral trigger source as task 8.

- **Enable IACK to start a task from software, if desired**

To enable the IACK instruction to start a task set the MCTL[IACKE] bit. Using the IACK instruction avoids having to set and clear the EALLOW bit. If the background task is enabled, the IACK bit for task 8 is ignored; the user must, instead, write to the BGSTART bit of the MCTLBGRND register to start the background task (TRIGEN should be 0 to avoid a peripheral trigger from causing an overflow, for example, MSTSBGRND.BGOVF is set to 1).

- **Map CLA data RAM to CLA space, if necessary**

Map the data RAM to the CLA space by first, assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL\_LSx] bit, and then specifying

the memory block as a CLA data block by writing a 0 to the MemCfgRegs.LSxCLAPGM[CLAPGM\_LSx] bit. When an LSx memory is configured as a CLA data memory, the CLA read/write accesses are arbitrated along with CPU accesses. The user has the option of turning on CPU fetch or write protection to the memory by writing to the appropriate bits of the MemCfgRegs.LSxACCPROT<sub>x</sub> registers.

- **Map CLA program RAM to CLA space**

Map the CLA program RAM to CLA space by first assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL\_LSx] bit, and then specifying the memory block as CLA code memory by writing a 1 to the MemCfgRegs.LSxCLAPGM[CLAPGM\_LSx] bit. When an LSx memory is configured as CLA program memory, only debug accesses are allowed on cycles in which the CLA is not fetching a new instruction.

#### 4. Initialize the PIE vector table and registers

When a CLA task completes, the associated interrupt in the PIE will be flagged. The CLA overflow and underflow flags also have associated interrupts within the PIE.

#### 5. Enable CLA tasks/interrupts

Set appropriate bits in the interrupt enable register (MIER) to allow the CLA to service interrupts.

#### 6. Initialize other peripherals

Initialize any peripherals (such as ePWM, ADC, and others) that will generate interrupt triggers for enabled CLA tasks.

The CLA is now ready to service interrupts and the message RAMs can be used to pass data between the CPU and the CLA. Mapping of the CLA program and data RAMs typically occurs only during the initialization process. If the RAM mapping needs to be changed after initialization, the CLA interrupts must be disabled and all tasks must be completed (by checking the MIRUN register) prior to modifying the RAM ownership.

### 5.5.3 Debugging CLA Code

Debugging the CLA code is a simple process that occurs independently of the main CPU. The type 2 CLA adds a true software breakpoint feature.

#### 5.5.3.1 Software Breakpoint Support (MDEBUGSTOP1)

The MDEBUGSTOP1 instruction is meant to be used as a software breakpoint; the instruction on which the execution must halt is replaced with this instruction.

The MDEBUGSTOP1 and MDEBUGSTOP instructions differ in how the CLA pipeline is treated. When halted, the MDEBUGSTOP1 instruction will flush all the instructions that have already been fetched; on a single-step or run-free command, the CLA will re-fetch the same instruction which it replaced. [Table 5-2](#) illustrates the pipeline behavior.

**Table 5-2. Pipeline Behavior of the MDEBUGSTOP1 Instruction**

Cycles	F1	F2	D1	D2	R1	R2	E	W	Comments
1	i1								
2	i2	i1							
3	i3	i2	i1						
4	i4	i3	i2	i1					
5	MDEBUGSTOP1	i4	i3	i2	i1				
6	i6	MDEBUGSTOP1	i4	i3	i2	i1			
7	i7	i6	MDEBUGSTOP1	i4	i3	i2	i1		
9	i8	Flushed (MNOP)	Flushed (MNOP)	Flushed (MNOP)	i4	i3	i2	i1	CLA halted

**Table 5-2. Pipeline Behavior of the MDEBUGSTOP1 Instruction (continued)**

Cycles	F1	F2	D1	D2	R1	R2	E	W	Comments
10	i5(MDEBU GSTOP1)	Flushed (MNOP)	Flushed (MNOP)	Flushed (MNOP)	Flushed (MNOP)	i4	i3	i2	CLA step/run
11	i6	i5(MDEBU GSTOP1)	Flushed (MNOP)	Flushed (MNOP)	Flushed (MNOP)	Flushed (MNOP)	i4	i3	CLA step/run
12	i7	i6	i5(MDEBU GSTOP1)	Flushed (MNOP)	Flushed (MNOP)	Flushed (MNOP)	Flushed (MNOP)	i4	CLA step/run
13	i8	i7	i6	i5(MDEBU GSTOP1)	Flushed (MNOP)	Flushed (MNOP)	Flushed (MNOP)	Flushed (MNOP)	CLA step/run

A software breakpoint is placed at instruction i5. The instruction, i5, is then replaced with MDEBUGSTOP1. It takes 3 cycles for the MDEBUGSTOP1 to reach the D2 phase at which point the instructions i6, i7, and i8 that were previously fetched are now flushed from the pipeline. The instruction, i5, is then re-fetched and execution continues as before.

### 5.5.3.2 Legacy Breakpoint Support (MDEBUGSTOP)

#### 1. Insert a breakpoint in CLA code

Insert a CLA breakpoint (MDEBUGSTOP instruction) into the code where you want the CLA to halt, then rebuild and reload the code. Because the CLA does not flush its pipeline when you single-step, the MDEBUGSTOP instruction must be inserted as part of the code. The debugger cannot insert it as needed.

If CLA breakpoints are not enabled, then the MDEBUGSTOP will be ignored and is treated as a MNOP. The MDEBUGSTOP instruction can be placed anywhere in the CLA code as long as it is not within three instructions of a MBCNDD, MCCNDD, or MRCNDD instruction. When programming in C, the user can use the `__mdebugstop()` intrinsic instead; the compiler will ensure that the placement of the MDEBUSTOP instruction in the generated assembly does not violate any of the pipeline restrictions.

#### 2. Enable CLA breakpoints

Enable the CLA breakpoints in the debugger. In Code Composer Studio, this is done by connecting to the CLA core (or tap) from the debug perspective. Breakpoints are disabled when the core is disconnected.

#### 3. Start the task

There are three ways to start the task:

1. The peripheral can assert an interrupt,
2. The main CPU can execute an IACK instruction, or
3. The user can manually write to the MIFRC register in the debugger window

When the task starts, the CLA will execute instructions until the MDEBUGSTOP is in the D2 phase of the pipeline. At this point, the CLA will halt and the pipeline will be frozen. The MPC register will reflect the address of the MDEBUGSTOP instruction.

#### 4. Single-step the CLA code

Once halted, the user can single-step the CLA code. The behavior of a CLA single-step is different than the main C28x. When issuing a CLA single-step, the pipeline is clocked only one cycle and then again frozen. On the 28x CPU, the pipeline is flushed for each single-step.

You can also run to the next MDEBUGSTOP or to the end of the task. If another task is pending, it will automatically start when you run to the end of the task.



**NOTE:** A CLA fetch has higher priority than CPU debug reads. For this reason, it is possible for the CLA to permanently block CPU debug accesses if the CLA is executing in a loop. This might occur when initially developing CLA code due to a bug that causes an infinite loop. To avoid locking up the main CPU, the program memory will return all 0x0000 for CPU debug reads when the CLA is running. When the CLA is halted or idle then normal CPU debug read and write access to CLA program memory can be performed.

If the CLA gets caught in an infinite loop, you can use a soft or hard reset to exit the condition. A debugger reset will also exit the condition.

There are special cases that can occur when single-stepping a task such that the program counter, MPC, reaches the MSTOP instruction at the end of the task.

- **MPC halts at or after the MSTOP with a task already pending**

If you are single-stepping or halted in "task A" and "task B" comes in before the MPC reaches the MSTOP, then "task B" will start if you continue to step through the MSTOP instruction. Basically if "task B" is pending before the MPC reaches MSTOP in "task A" then there is no issue in "task B" starting and no special action is required.

- **MPC halts at or after the MSTOP with no task pending**

In this case you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. If "task B" comes in at this point, it will be flagged in the MIFR register but it may or may not start if you continue to single-step through the MSTOP instruction of "task A."

It depends on exactly when the new task comes in. To reliably start "task B" perform a soft reset and reconfigure the MIER bits. Once this is done, you can start single-stepping "task B."

This case can be handled slightly differently if there is control over when "task B" comes in (for example using the IACK instruction to start the task). In this case you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. Before forcing "task B," run free to force the CLA out of the debug state. Once this is done you can force "task B" and continue debugging.

#### 5. Disable CLA breakpoints, if desired

In Code Composer Studio you can disable the CLA breakpoints by disconnecting the CLA core in the debug perspective. Make sure to first issue a run or reset; otherwise, the CLA will be halted and no other tasks will start.

### 5.5.4 CLA Illegal Opcode Behavior

If the CLA fetches an opcode that does not correspond to a legal instruction, it will behave as follows:

- The CLA will halt with the illegal opcode in the D2 phase of the pipeline as if it were a breakpoint. This will occur whether CLA breakpoints are enabled or not.
- The CLA will issue the task-specific interrupt to the PIE.
- The MIRUN bit for the task will remain set.

Further single-stepping is ignored once execution halts due to an illegal op-code. To exit this situation, issue either a soft or hard reset of the CLA as described in [Section 5.5.5](#).

### 5.5.5 Resetting the CLA

There may be times when you need to reset the CLA. For example, during code debug the CLA may enter an infinite loop due to a code bug. The CLA has two types of resets: hard and soft. Both of these resets can be performed by the debugger or by the main CPU.

- **Hard Reset**

Writing a 1 to the MCTL[HARDRESET] bit will perform a hard reset of the CLA. The behavior of a hard reset is the same as a system reset (via  $\overline{\text{XRS}}$  or the debugger). In this case all CLA configuration and execution registers will be set to their default state and CLA execution will halt.

- **Soft Reset**

Writing a 1 to the MCTL[SOFTRESET] bit performs a soft reset of the CLA. If a task is executing it will halt and the associated MIRUN bit will be cleared. All bits within the interrupt enable (MIER) register

will also be cleared so that no new tasks start. In addition, the background task's start bit (MCTLBGRN.BGSTART) and Trigger Enable bit (MCTLBGRND.TRIGEN) are reset. The MVECTBGRNACTIVE is set to the value of MVECTBGRND, and the status register (MSTSBGRND) is also reset.



## 5.6 Pipeline

This section describes the CLA pipeline stages and presents cases where pipeline alignment must be considered.

### 5.6.1 Pipeline Overview

The CLA pipeline is very similar to the C28x pipeline with eight stages:

1. **Fetch 1 (F1)**  
During the F1 stage the program read address is placed on the CLA program address bus.
2. **Fetch 2 (F2)**  
During the F2 stage the instruction is read using the CLA program data bus.
3. **Decode 1 (D1)**  
During D1 the instruction is decoded.
4. **Decode 2 (D2)**  
Generate the data read address. Changes to MAR0 and MAR1 due to post-increment using indirect addressing takes place in the D2 phase. Conditional branch decisions are also made at this stage based on the MSTF register flags.
5. **Read 1 (R1)**  
Place the data read address on the CLA data-read address bus. If a memory conflict exists, the R1 stage will be stalled.
6. **Read 2 (R2)**  
Read the data value using the CLA data read data bus.
7. **Execute (EXE)**  
Execute the operation. Changes to MAR0 and MAR1 due to loading an immediate value or value from memory take place in this stage.
8. **Write (W)**  
Place the write address and write data on the CLA write data bus. If a memory conflict exists, the W stage will be stalled.

### 5.6.2 CLA Pipeline Alignment

The majority of the CLA instructions do not require any special pipeline considerations. This section lists the few operations that do require special consideration.

- **Write Followed by Read**

In both the C28x and the CLA pipeline the read operation occurs before the write. This means that if a read operation immediately follows a write, then the read will complete first as shown in [Table 5-3](#). In most cases this does not cause a problem since the contents of one memory location does not depend on the state of another. For accesses to peripherals where a write to one location can affect the value in another location the code must wait for the write to complete before issuing the read as shown in [Table 5-4](#).

This behavior is different for the 28x CPU. For the 28x CPU any write followed by read to the same location is protected by what is called write-followed-by-read protection. This protection automatically stalls the pipeline so that the write will complete before the read. In addition some peripheral frames are protected such that a 28x CPU write to one location within the frame will always complete before a read to the frame. The CLA does not have this protection mechanism. Instead the code must wait to perform the read.

**Table 5-3. Write Followed by Read - Read Occurs First**

Instruction	F1	F2	D1	D2	R1	R2	E	W
I1 MMOV16 @Reg1, MR3	I1							
I2 MMOV16 MR2, @Reg2	I2	I1						
		I2	I1					
			I2	I1				
				I2	I1			
					I2	I1		
						I2	I1	
							I2	I1

**Table 5-4. Write Followed by Read - Write Occurs First**

Instruction	F1	F2	D1	D2	R1	R2	E	W
I1 MMOV16 @Reg1, MR3	I1							
I2	I2	I1						
I3	I3	I2	I1					
I4	I4	I3	I2	I1				
I5 MMOV16 MR2, @Reg2	I5	I4	I3	I2	I1			
		I5	I4	I3	I2	I1		
			I5	I4	I3	I2	I1	
				I5	I4	I3	I2	I1
					I5	I4	I3	I1
						I5	I4	I1
							I5	I1

- **Delayed Conditional instructions: MBCNDD, MCCNDD and MRCNDD**

Referring to [Example 5-1](#), the following applies to delayed conditional instructions:

- **I1**

I1 is the last instruction that can effect the CNDF flags for the branch, call or return instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to branch or not when [MBCNDD](#), [MCCNDD](#) or [MRCNDD](#) is in the D2 phase.

- **I2, I3 and I4**

The three instructions preceding [MBCNDD](#) can change MSTF flags but will have no effect on whether the [MBCNDD](#) instruction branches or not. This is because the flag modification will occur after the D2 phase of the branch, call or return instruction. These three instructions must not be a [MSTOP](#), [MDEBUGSTOP](#), [MBCNDD](#), [MCCNDD](#) or [MRCNDD](#).

- **I5, I6 and I7**

The three instructions following a branch, call or return are always executed irrespective of whether the condition is true or not. These instructions must not be [MSTOP](#), [MDEBUGSTOP](#), [MBCNDD](#), [MCCNDD](#) or [MRCNDD](#).

For a more detailed description refer to the functional description for [MBCNDD](#), [MCCNDD](#) and [MRCNDD](#).

**Example 5-1. Code Fragment For MBCNDD, MCCNDD or MRCNDD**

```

<Instruction 1> ; I1 Last instruction that can affect flags for
                ; the branch, call or return operation

<Instruction 2> ; I2 Cannot be stop, branch, call or return
<Instruction 3> ; I3 Cannot be stop, branch, call or return
<Instruction 4> ; I4 Cannot be stop, branch, call or return

<branch/call/ret> ; MBCNDD, MCCNDD or MRCNDD

                ; I5-I7: Three instructions after are always
                ; executed whether the branch/call or return is
                ; taken or not

<Instruction 5> ; I5 Cannot be stop, branch, call or return
<Instruction 6> ; I6 Cannot be stop, branch, call or return
<Instruction 7> ; I7 Cannot be stop, branch, call or return

<Instruction 8> ; I8
<Instruction 9> ; I9
....

```

- **Stop or Halting a Task: MSTOP and MDEBUGSTOP**

The [MSTOP](#) and [MDEBUGSTOP](#) instructions cannot be placed three instructions before or after a conditional branch, call or return instruction ([MBCNDD](#), [MCCNDD](#) or [MRCNDD](#)). Refer to [Example 5-1](#). To single-step through a branch/call or return, insert the [MDEBUGSTOP](#) at least four instructions back and step from there.

- **Loading MAR0 or MAR1**

A load of auxiliary register MAR0 or MAR1 will occur in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing will occur in the D2 phase of the pipeline. Referring to [Example 5-2](#), the following applies when loading the auxiliary registers:

- **I1 and I2**

The two instructions following the load instruction will use the value in MAR0 or MAR1 before the update occurs.

- **I3**

Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus I3 cannot use the auxiliary register or there will be a conflict. In the case of a conflict, the update due to address-mode post increment will win and the auxiliary register will not be updated with #\_X.

- **I4**

Starting with the 4th instruction MAR0 or MAR1 will have the new value.

- **Background Task Interrupted Close to a Branch**

When the background task is running, if another task request happens (and `MSTSBGRND.BGINTM` is not set), then the following sequence of operations happen.

- A check is made to determine if the following instructions are not in the pipeline (D2 – R2).

- [MBCNDD](#)
- [MCCNDD](#)
- [MRCNDD](#)

If any of the above instructions are present in the pipeline, the back ground task continues to execute until such time when the condition is satisfied. Once it is satisfied, the following actions are performed:

- The MPC value of the D1 phase instruction is saved to the `MVECTBGRNDACTIVE` register.
- The run status bit of the background task (`MSTSBGRND.RUNSTS`) is cleared.
- An `MSTOP` instruction is forced into the D2 phase of the pipeline; it will cause the background task

to terminate.

When the background task terminates, the CLA picks the next highest pending task and begins execution. It is important to note that while the background task is pending it has the lowest priority and will, therefore, yield to any other pending task. Once all pending non-background tasks have completed execution the CLA will restore the program counter (MPC), that is, load the address from the MVECTBGRNDACTIVE register to the MPC, set the background status to RUN (MSTSBGRND.RUN = 1), and continue execution from that point.

- **MSTOP in the Background Task**

If an MSTOP instruction occurs in the D1 phase while the background task is running, the following sequence of operations happens:

- The RUN bit (MSTSBGRND.RUN) is cleared.
- The address stored in MVECTBGRND is copied over to MVECTBGRNDACTIVE.
- An interrupt, signaling the background task has completed execution, is generated. This interrupt signal is ANDed with the interrupt from Task 8 and fed to the PIE. It should be noted that if an illegal instruction occurs inside the background task, the interrupt for task 8 is fired.
- When the background task terminates, the CLA resumes arbitration among the pending tasks.

**Example 5-2. Code Fragment for Loading MAR0 or MAR1**

```

; Assume MAR0 is 50 and #_X is 20

MMOV16 MAR0, #_X ; Load MAR0 with address of X (20)
<Instruction 1> ; I1 Will use the old value of MAR0 (50)
<Instruction 2> ; I2 Will use the old value of MAR0 (50)
<Instruction 3> ; I3 Cannot use MAR0
<Instruction 4> ; I4 Will use the new value of MAR0 (20)
<Instruction 5> ; I5 Will use the new value of MAR0 (20)
....

```

**5.6.2.1 ADC Early Interrupt to CLA Response**

The ADC can be configured to generate an early interrupt pulse before the ADC conversion completes. If this option is used to start a CLA task, the CLA will be able to read the result as soon as the conversion result is available in the ADC result register. This combination of just-in-time sampling along with the low interrupt response of the CLA enable faster system response and higher frequency control loops.

Timings for ADC conversions are shown in the timing diagrams of the ADC chapter. If the ADCCLK is a divided down version of the SYSCLK, the user will have to account for the conversion time in SYSCLK cycles.

For example, if using the 12-bit ADC with ADCCLK at SYSCLK / 4, it would take 10.5 ADCCLK x 4 SYSCLK = 42 SYSCLK cycles to complete a conversion.

From a CLA perspective, the pipeline activity is shown in [Table 5-5](#) for an N-cycle (SYSCLK) ADC conversion. The N-2 instruction will arrive in the R2 phase just in time to read the result register. While the prior instructions will enter the R2 phase of the pipeline too soon to read the conversion, they can be efficiently used for pre-processing calculations needed by the task.

**Table 5-5. ADC to CLA Early Interrupt Response**

ADC Activity	CLA Activity	F1	F2	D1	D2	R1	R2	E	W
Sample									
Sample									
...									
Sample									
Conversion <sub>(Cycle 1)</sub>	Interrupt Received								
Conversion <sub>(Cycle 2)</sub>	Task Startup								
Conversion <sub>(Cycle 3)</sub>	Task Startup								

**Table 5-5. ADC to CLA Early Interrupt Response (continued)**

ADC Activity	CLA Activity	F1	F2	D1	D2	R1	R2	E	W
Conversion <sub>(Cycle 4)</sub>	I <sub>(Cycle 4)</sub>	I <sub>(Cycle 4)</sub>							
Conversion <sub>(Cycle 5)</sub>	I <sub>(Cycle 5)</sub>	I <sub>(Cycle 5)</sub>	I <sub>(Cycle 4)</sub>						
Conversion <sub>(...)</sub>	...	...	...	...	...	...	...		
Conversion <sub>(Cycle N-6)</sub>	I <sub>(Cycle N-6)</sub>	I <sub>(Cycle N-6)</sub>	I <sub>(Cycle N-7)</sub>	I <sub>(Cycle N-8)</sub>	I <sub>(Cycle N-9)</sub>	I <sub>(Cycle N-10)</sub>	I <sub>(Cycle N-11)</sub>		
Conversion <sub>(Cycle N-5)</sub>	I <sub>(Cycle N-5)</sub>	I <sub>(Cycle N-5)</sub>	I <sub>(Cycle N-6)</sub>	I <sub>(Cycle N-7)</sub>	I <sub>(Cycle N-8)</sub>	I <sub>(Cycle N-9)</sub>	I <sub>(Cycle N-10)</sub>		
Conversion <sub>(Cycle N-4)</sub>	I <sub>(Cycle N-4)</sub>	I <sub>(Cycle N-4)</sub>	I <sub>(Cycle N-5)</sub>	I <sub>(Cycle N-6)</sub>	I <sub>(Cycle N-7)</sub>	I <sub>(Cycle N-8)</sub>	I <sub>(Cycle N-9)</sub>		
Conversion <sub>(Cycle N-3)</sub>	I <sub>(Cycle N-3)</sub>	I <sub>(Cycle N-3)</sub>	I <sub>(Cycle N-4)</sub>	I <sub>(Cycle N-5)</sub>	I <sub>(Cycle N-6)</sub>	I <sub>(Cycle N-7)</sub>	I <sub>(Cycle N-8)</sub>		
Conversion <sub>(Cycle N-2)</sub>	<b>Read RESULT</b>	<b>Read RESULT</b>	I <sub>(Cycle N-3)</sub>	I <sub>(Cycle N-4)</sub>	I <sub>(Cycle N-5)</sub>	I <sub>(Cycle N-6)</sub>	I <sub>(Cycle N-7)</sub>		
Conversion <sub>(Cycle N-1)</sub>			<b>Read RESULT</b>	I <sub>(Cycle N-3)</sub>	I <sub>(Cycle N-4)</sub>	I <sub>(Cycle N-5)</sub>	I <sub>(Cycle N-6)</sub>		
Conversion <sub>(Cycle N-0)</sub>				<b>Read RESULT</b>	I <sub>(Cycle N-3)</sub>	I <sub>(Cycle N-4)</sub>	I <sub>(Cycle N-5)</sub>		
Conversion Complete					<b>Read RESULT</b>	I <sub>(Cycle N-3)</sub>	I <sub>(Cycle N-4)</sub>		
RESULT Latched						<b>Read RESULT</b>	I <sub>(Cycle N-3)</sub>		
RESULT Available							<b>Read RESULT</b>		

### 5.6.3 Parallel Instructions

Parallel instructions are single opcodes that perform two operations in parallel. The following types of parallel instructions are available: math operation in parallel with a move operation, or two math operations in parallel. Both operations complete in a single cycle and there are no special pipeline alignment requirements.

#### Example 5-3. Math Operation with Parallel Load

```

; MADDF32 || MMOV32 instruction: 32-bit floating-point add with parallel move
; MADDF32 is a 1 cycle operation
; MMOV32 is a 1 cycle operation
    MADDF32    MR0, MR1, #2    ; MR0 = MR1 + 2,
    || MMOV32  MR1, @Val      ; MR1 gets the contents of Val
                                ; <-- MMOV32 completes here (MR1 is valid)
                                ; <-- DDF32 completes here (MR0 is valid)
    MMPYF32 MR0, MR0, MR1    ; Any instruction, can use MR1 and/or MR0
    
```

#### Example 5-4. Multiply with Parallel Add

```

; MMPYF32 || MADDF32 instruction: 32-bit floating-point multiply with parallel add
; MMPYF32 is a 1 cycle operation
; MADDF32 is a 1 cycle operation
    MMPYF32 MR0, MR1, MR3    ; MR0 = MR1 * MR3
    || MADDF32 MR1, MR2, MR0 ; MR1 = MR2 + MR0 (Uses value of MR0 before MMPYF32)
                                ; <-- MMPYF32 and MADDF32 complete here (MR0 and MR1 are valid)
    MMPYF32 MR1, MR1, MR0    ; Any instruction, can use MR1 and/or MR0
    
```

## 5.7 Instruction Set

This section describes the assembly language instructions of the control law accelerator. Also described are parallel operations, conditional operations, resource constraints, and addressing modes. The instructions listed here are independent from C28x and C28x+FPU instruction sets.

### 5.7.1 Instruction Descriptions

This section gives detailed information on the instruction set. Each instruction may present the following information:

- Operands
- Opcode
- Description
- Exceptions
- Pipeline
- Examples
- See also

The example INSTRUCTION is shown to familiarize you with the way each instruction is described. The example describes the kind of information you will find in each part of the individual instruction description and where to obtain more information. CLA instructions follow the same format as the C28x; the source operand(s) are always on the right and the destination operand(s) are on the left.

The explanations for the syntax of the operands used in the instruction descriptions for the C28x CLA are given in [Table 5-6](#).

**Table 5-6. Operand Nomenclature**

Symbol	Description
#16FHi	16-bit immediate (hex or float) value that represents the upper 16-bits of an IEEE 32-bit floating-point value. Lower 16-bits of the mantissa are assumed to be zero.
#16FHiHex	16-bit immediate hex value that represents the upper 16-bits of an IEEE 32-bit floating-point value. Lower 16-bits of the mantissa are assumed to be zero.
#16FLoHex	A 16-bit immediate hex value that represents the lower 16-bits of an IEEE 32-bit floating-point value
#32Fhex	32-bit immediate value that represents an IEEE 32-bit floating-point value
#32F	Immediate float value represented in floating-point representation
#0.0	Immediate zero
#SHIFT	Immediate value of 1 to 32 used for arithmetic and logical shifts.
addr	Opcode field indicating the addressing mode
CNDF	Condition to test the flags in the MSTF register
FLAG	Selected flags from MSTF register (OR) 8 bit mask indicating which floating-point status flags to change
MAR0	auxiliary register 0
MAR1	auxiliary register 1
MARx	Either MAR0 or MAR1
mem16	16-bit memory location accessed using direct, indirect, or offset addressing modes
mem32	32-bit memory location accessed using direct, indirect, or offset addressing modes
MRa	MR0 to MR3 registers
MRb	MR0 to MR3 registers
MRc	MR0 to MR3 registers
MRd	MR0 to MR3 registers
MRe	MR0 to MR3 registers
MRf	MR0 to MR3 registers
MSTF	CLA Floating-point Status Register
shift	Opcode field indicating the number of bits to shift.
VALUE	Flag value of 0 or 1 for selected flag (OR) 8 bit mask indicating the flag value; 0 or 1

Each instruction has a table that gives a list of the operands and a short description. Instructions always have their destination operand(s) first followed by the source operand(s).

**Table 5-7. INSTRUCTION dest, source1, source2 Short Description**

	Description
dest1	Description for the 1st operand for the instruction
source1	Description for the 2nd operand for the instruction
source2	Description for the 3rd operand for the instruction
Opcode	This section shows the opcode for the instruction
Description	Detailed description of the instruction execution is described. Any constraints on the operands imposed by the processor or the assembler are discussed.
Restrictions	Any constraints on the operands or use of the instruction imposed by the processor are discussed.
Pipeline	This section describes the instruction in terms of pipeline cycles as described in <a href="#">Section 5.6</a>
Example	Examples of instruction execution. If applicable, register and memory values are given before and after instruction execution. Some examples are code fragments while other examples are full tasks that assume the CLA is correctly configured and the main CPU has passed it data.
Operands	Each instruction has a table that gives a list of the operands and a short description. Instructions always have their destination operand(s) first followed by the source operand(s).



## 5.7.2 Addressing Modes and Encoding

The CLA uses the same address to access data and registers as the main CPU. For example if the main CPU accesses an ePWM register at address 0x00 6800, then the CLA will access it using address 0x6800. Since all CLA accessible memory and registers are within the low 64k x 16 of memory, only the low 16-bits of the address are used by the CLA.

To address the CLA data memory, message RAMs and shared peripherals, the CLA supports two addressing modes:

- Direct addressing mode: Uses the address of the variable or register directly.
- Indirect addressing with 16-bit post increment. This mode uses either XAR0 or XAR1.

The CLA does not use a data page pointer or a stack pointer. The two addressing modes are encoded as shown [Table 5-8](#).

**Table 5-8. Addressing Modes**

Addressing Mode	'addr' Opcode Field Encode <sup>(1)</sup>	Description
@dir	0000	<p><b>Direct Addressing Mode</b></p> <p>Example 1: MMOV32 MR1, @_VarA</p> <p>Example 2: MMOV32 MR1, @_EPwm1Regs.CMPA.all</p> <p>In this case the 'm m m m m m m m m m m m m m m m' opcode field will be populated with the 16-bit address of the variable. This is the low 16-bits of the address that you would use to access the variable using the main CPU.</p> <p>For example @_VarA will populate the address of the variable VarA. and @_EPwm1Regs.CMPA.all will populate the address of the CMPA register.</p>
*MAR0[#imm16]++	0001	<p><b>MAR0 Indirect Addressing with 16-bit Immediate Post Increment</b></p>
*MAR1[#imm16]++	0010	<p><b>MAR1 Indirect Addressing with 16-bit Immediate Post Increment</b></p> <p>addr = MAR0 (or MAR1) Access memory using the address stored in MAR0 (or MAR1).  MAR0 (or MAR1) += Then post increment MAR0 (or MAR1) by #imm16.  #imm16</p> <p>Example 1: MMOV32 MR0, *MAR0[2]++</p> <p>Example 2: MMOV32 MR1, *MAR1[-2]++</p> <p>For a post increment of 0 the assembler will accept both *MAR0 and *MAR0[0]++.</p> <p>The 'm m m m m m m m m m m m m m m m' opcode field will be populated with the signed 16-bit pointer offset. For example if #imm16 is 2, then the opcode field will be 0x0002. Likewise if #imm16 is -2, then the opcode field will be 0xFFFF.</p> <p>If addition of the 16-bit immediate causes overflow, then the value will wrap around on a 16-bit boundary.</p>
*MAR0+[#imm16]	0101	<p><b>MAR0 Offset Addressing with 16-bit Immediate Offset</b></p>
*MAR1+[#imm16]	0110	<p><b>MAR1 Offset Addressing with 16-bit Immediate Offset</b></p> <p>addr = MAR0 Add the offset #imm16  (or MAR1) + #imm16to address stored in MAR0(MAR1) to access the desired memory  the base location</p> <p>Example 1: MMOV32 MR0, *MAR0+[2]</p> <p>Example 1: MMOV32 MR1, *MAR1+[-2]</p> <p>The 'm m m m m m m m m m m m m m m m' opcode field will be populated with the signed 16-bit pointer offset. For example if #imm16 is 2, then the opcode field will be 0x0002. Likewise if #imm16 is -2, then the opcode field will be 0xFFFF.</p> <p>If the addition of the 16-bit immediate causes overflow, the value will wrap around on a 16-bit boundary.</p>

<sup>(1)</sup> Values not shown are reserved.

Encoding for the shift fields in the MASR32, MLRS32 and MLSL32 instructions is shown in [Table 5-9](#).

**Table 5-9. Shift Field Encoding**

Shift Value	'shift' Opcode Field Encode
1	0000
2	0001
3	0010
....	....
32	1111

Table 5-11 shows the condition field encoding for conditional instructions such as MNEGF, MSWAPF, MBCNDD, MCCNDD, and MRCNDD.

For instructions that use MRx (where x could be 'a' through 'f') as operands, the trailing alphabet appears in the opcode as a two-bit field. For example,

```
MMPYF32 MRa, MRb, MRc ||
MADDF32 MRd, MRe, MRf
```

whose opcode is,

```
LSW: 0000 ffee dccc bbaa
MSW: 0111 1010 0000 0000
```

The two-bit field specifies one of four working registers according to Table 5-10.

**Table 5-10. Operand Encoding**

Two-Bit Field	Working Register
b'00	MR0
b'01	MR1
b'10	MR2
b'11	MR3

**Table 5-11. Condition Field Encoding**

Encode <sup>(1)</sup>	CNDF	Description	MSTF Flags Tested
0000	NEQ	Not equal to zero	ZF == 0
0001	EQ	Equal to zero	ZF == 1
0010	GT	Greater than zero	ZF == 0 AND NF == 0
0011	GEQ	Greater than or equal to zero	NF == 0
0100	LT	Less than zero	NF == 1
0101	LEQ	Less than or equal to zero	ZF == 1 OR NF == 1
1010	TF	Test flag set	TF == 1
1011	NTF	Test flag not set	TF == 0
1100	LU	Latched underflow	LUF == 1
1101	LV	Latched overflow	LVF == 1
1110	UNC	Unconditional	None
1111	UNCF <sup>(2)</sup>	Unconditional with flag modification	None

<sup>(1)</sup> Values not shown are reserved.

<sup>(2)</sup> This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

### 5.7.3 Instructions

The instructions are listed alphabetically, preceded by a summary.

**Table 5-12. General Instructions**

Title	Page
<b>MMABSF32 MRa, MRb</b> — 32-Bit Floating-Point Absolute Value.....	669
<b>MADD32 MRa, MRb, MRc</b> — 32-Bit Integer Add.....	670
<b>MADDF32 MRa, #16FHi, MRb</b> — 32-Bit Floating-Point Addition.....	671
<b>MADDF32 MRa, MRb, #16FHi</b> — 32-Bit Floating-Point Addition.....	672
<b>MADDF32 MRa, MRb, MRc</b> — 32-Bit Floating-Point Addition.....	674
<b>MADDF32 MRd, MRe, MRf   MMOV32 mem32, MRa</b> — 32-Bit Floating-Point Addition with Parallel Move.....	675
<b>MADDF32 MRd, MRe, MRf   MMOV32 MRa, mem32</b> — 32-Bit Floating-Point Addition with Parallel Move.....	676
<b>MAND32 MRa, MRb, MRc</b> — Bitwise AND.....	678
<b>MASR32 MRa, #SHIFT</b> — Arithmetic Shift Right.....	679
<b>MBCNDD 16BitDest {, CNDF}</b> — Branch Conditional Delayed.....	680
<b>MCCNDD 16BitDest {, CNDF}</b> — Call Conditional Delayed.....	685
<b>MCLRC BGINTM</b> — Clear Background Task Interrupt Mask.....	689
<b>MCMP32 MRa, MRb</b> — 32-Bit Integer Compare for Equal, Less Than or Greater Than.....	690
<b>MCMPF32 MRa, MRb</b> — 32-Bit Floating-Point Compare for Equal, Less Than or Greater Than.....	691
<b>MCMPF32 MRa, #16FHi</b> — 32-Bit Floating-Point Compare for Equal, Less Than or Greater Than.....	692
<b>MDEBUGSTOP</b> — Debug Stop Task.....	694
<b>MDEBUGSTOP1</b> — Software Breakpoint.....	695
<b>MEALLOW</b> — Enable CLA Write Access to EALLOW Protected Registers.....	696
<b>MEDIS</b> — Disable CLA Write Access to EALLOW Protected Registers.....	697
<b>MEINVF32 MRa, MRb</b> — 32-Bit Floating-Point Reciprocal Approximation.....	698
<b>MEISQRTF32 MRa, MRb</b> — 32-Bit Floating-Point Square-Root Reciprocal Approximation.....	699
<b>MF32TOI16 MRa, MRb</b> — Convert 32-Bit Floating-Point Value to 16-Bit Integer.....	700
<b>MF32TOI16R MRa, MRb</b> — Convert 32-Bit Floating-Point Value to 16-Bit Integer and Round.....	701
<b>MF32TOI32 MRa, MRb</b> — Convert 32-Bit Floating-Point Value to 32-Bit Integer.....	702
<b>MF32TOUI16 MRa, MRb</b> — Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer.....	703
<b>MF32TOUI16R MRa, MRb</b> — Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer and Round.....	704
<b>MF32TOUI32 MRa, MRb</b> — Convert 32-Bit Floating-Point Value to 32-Bit Unsigned Integer.....	705
<b>MFRACF32 MRa, MRb</b> — Fractional Portion of a 32-Bit Floating-Point Value.....	706
<b>MI16TOF32 MRa, MRb</b> — Convert 16-Bit Integer to 32-Bit Floating-Point Value.....	707
<b>MI16TOF32 MRa, mem16</b> — Convert 16-Bit Integer to 32-Bit Floating-Point Value.....	708
<b>MI32TOF32 MRa, mem32</b> — Convert 32-Bit Integer to 32-Bit Floating-Point Value.....	709
<b>MI32TOF32 MRa, MRb</b> — Convert 32-Bit Integer to 32-Bit Floating-Point Value.....	710
<b>MLSL32 MRa, #SHIFT</b> — Logical Shift Left.....	711
<b>MLSR32 MRa, #SHIFT</b> — Logical Shift Right.....	712
<b>MMACF32 MR3, MR2, MRd, MRe, MRf   MMOV32 MRa, mem32</b> — 32-Bit Floating-Point Multiply and Accumulate with Parallel Move.....	713
<b>MMAXF32 MRa, MRb</b> — 32-Bit Floating-Point Maximum.....	716
<b>MMAXF32 MRa, #16FHi</b> — 32-Bit Floating-Point Maximum.....	718
<b>MMINF32 MRa, MRb</b> — 32-Bit Floating-Point Minimum.....	719
<b>MMINF32 MRa, #16FHi</b> — 32-Bit Floating-Point Minimum.....	721
<b>MMOV16 MARx, MRa, #16I</b> — Load the Auxiliary Register with MRa + 16-bit Immediate Value.....	722
<b>MMOV16 MARx, mem16</b> — Load MAR1 with 16-bit Value.....	725
<b>MMOV16 mem16, MARx</b> — Move 16-Bit Auxiliary Register Contents to Memory.....	727
<b>MMOV16 mem16, MRa</b> — Move 16-Bit Floating-Point Register Contents to Memory.....	728
<b>MMOV32 mem32, MRa</b> — Move 32-Bit Floating-Point Register Contents to Memory.....	730
<b>MMOV32 mem32, MSTF</b> — Move 32-Bit MSTF Register to Memory.....	731

**Table 5-12. General Instructions (continued)**

<b>MMOV32 MRa, mem32 {, CNDF}</b> —Conditional 32-Bit Move .....	732
<b>MMOV32 MRa, MRb {, CNDF}</b> —Conditional 32-Bit Move .....	734
<b>MMOV32 MSTF, mem32</b> —Move 32-Bit Value from Memory to the MSTF Register .....	736
<b>MMOVD32 MRa, mem32</b> —Move 32-Bit Value from Memory with Data Copy .....	737
<b>MMOV32 MRa, #32F</b> —Load the 32-Bits of a 32-Bit Floating-Point Register .....	738
<b>MMOVI16 MARx, #16I</b> —Load the Auxiliary Register with the 16-Bit Immediate Value .....	739
<b>MMOV32 MRa, #32FHex</b> —Load the 32-Bits of a 32-Bit Floating-Point Register with the Immediate .....	740
<b>MMOVIZ MRa, #16FHi</b> —Load the Upper 16-Bits of a 32-Bit Floating-Point Register .....	741
<b>MMOVZ16 MRa, mem16</b> —Load MRx With 16-bit Value .....	742
<b>MMOVXI MRa, #16FLoHex</b> —Move Immediate to the Low 16-Bits of a Floating-Point Register .....	743
<b>MMPYF32 MRa, MRb, MRc</b> —32-Bit Floating-Point Multiply .....	744
<b>MMPYF32 MRa, #16FHi, MRb</b> —32-Bit Floating-Point Multiply .....	745
<b>MMPYF32 MRa, MRb, #16FHi</b> —32-Bit Floating-Point Multiply .....	747
<b>MMPYF32 MRa, MRb, MRc   MADF32 MRd, MRe, MRf</b> —32-Bit Floating-Point Multiply with Parallel Add .....	749
<b>MMPYF32 MRd, MRe, MRf   MMOV32 MRa, mem32</b> —32-Bit Floating-Point Multiply with Parallel Move .....	751
<b>MMPYF32 MRd, MRe, MRf   MMOV32 mem32, MRa</b> —32-Bit Floating-Point Multiply with Parallel Move .....	753
<b>MMPYF32 MRa, MRb, MRc   MSUBF32 MRd, MRe, MRf</b> —32-Bit Floating-Point Multiply with Parallel Subtract .....	754
<b>MNEGF32 MRa, MRb{, CNDF}</b> —Conditional Negation .....	755
<b>MNOP</b> —No Operation .....	757
<b>MOR32 MRa, MRb, MRc</b> —Bitwise OR .....	758
<b>MRCNDD {CNDF}</b> —Return Conditional Delayed .....	759
<b>MSETC BGINTM</b> —Set Background Task Interrupt Mask .....	763
<b>MSETFLG FLAG, VALUE</b> —Set or Clear Selected Floating-Point Status Flags .....	764
<b>MSTOP</b> —Stop Task .....	765
<b>MSUB32 MRa, MRb, MRc</b> —32-Bit Integer Subtraction .....	767
<b>MSUBF32 MRa, MRb, MRc</b> —32-Bit Floating-Point Subtraction .....	768
<b>MSUBF32 MRa, #16FHi, MRb</b> —32-Bit Floating-Point Subtraction .....	769
<b>MSUBF32 MRd, MRe, MRf   MMOV32 MRa, mem32</b> —32-Bit Floating-Point Subtraction with Parallel Move .....	770
<b>MSUBF32 MRd, MRe, MRf   MMOV32 mem32, MRa</b> —32-Bit Floating-Point Subtraction with Parallel Move .....	771
<b>MSWAPF MRa, MRb {, CNDF}</b> —Conditional Swap .....	772
<b>MTESTTF CNDF</b> —Test MSTF Register Flag Condition .....	774
<b>MUI16TOF32 MRa, mem16</b> —Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value .....	776
<b>MUI16TOF32 MRa, MRb</b> —Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value .....	777
<b>MUI32TOF32 MRa, mem32</b> —Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value .....	778
<b>MUI32TOF32 MRa, MRb</b> —Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value .....	779
<b>MXOR32 MRa, MRb, MRc</b> —Bitwise Exclusive Or .....	780

## MMABSF32 MRa, MRb 32-Bit Floating-Point Absolute Value

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

### Opcode

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 0010 0000
```

### Description

The absolute value of MRb is loaded into MRa. Only the sign bit of the operand is modified by the MMABSF32 instruction.

```
if (MRb < 0) {MRa = -MRb};
else {MRa = MRb};
```

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified as follows:

```
NF = 0;
ZF = 0;
if ( MRa(30:23) == 0) ZF = 1;
```

### Pipeline

This is a single-cycle instruction.

### Example

```
MMOVIZ MR0, #-2.0 ; MR0 = -2.0 (0xC0000000)
MMABSF32 MR0, MR0 ; MR0 = 2.0 (0x40000000), ZF = NF = 0
```

```
MMOVIZ MR0, #5.0 ; MR0 = 5.0 (0x40A00000)
MMABSF32 MR0, MR0 ; MR0 = 5.0 (0x40A00000), ZF = NF = 0
```

```
MMOVIZ MR0, #0.0 ; MR0 = 0.0
MMABSF32 MR0, MR0 ; MR0 = 0.0 ZF = 1, NF = 0
```

### See also

[MNEGF32 MRa, MRb {, CNDF}](#)

**MADD32 MRa, MRb, MRc 32-Bit Integer Add**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point destination register (MR0 to MR3)
MRc	CLA floating-point destination register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 000cc bbaa
MSW: 0111 1110 1100 0000
```

**Description**

32-bit integer addition of MRb and MRc.

```
MRa(31:0) = MRb(31:0) + MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; };
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given A = (int32)1
;      B = (int32)2
;      C = (int32)-7
;
; Calculate Y2 = A + B + C
;
_ClalTask1:
    MMOV32 MR0, @_A      ; MR0 = 1 (0x00000001)
    MMOV32 MR1, @_B      ; MR1 = 2 (0x00000002)
    MMOV32 MR2, @_C      ; MR2 = -7 (0xFFFFFFFF9)
    MADD32 MR3, MR0, MR1 ; A + B
    MADD32 MR3, MR2, MR3 ; A + B + C = -4 (0xFFFFFFFFC)
    MMOV32 @_y2, MR3     ; Store y2
    MSTOP                ; end of task
```

**See also**

[MAND32 MRa, MRb, MRc](#)  
[MASR32 MRa, #SHIFT](#)  
[MLSL32 MRa, #SHIFT](#)  
[MLSR32 MRa, #SHIFT](#)  
[MOR32 MRa, MRb, MRc](#)  
[MXOR32 MRa, MRb, MRc](#)  
[MSUB32 MRa, MRb, MRc](#)

## MADDF32 MRa, #16FHi, MRb 32-Bit Floating-Point Addition

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
#16FHi	A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0.
MRb	CLA floating-point source register (MR0 to MR3)

### Opcode

```
LSW: I I I I I I I I I I I I I I I I
MSW: 0 1 1 1 0 1 1 1 1 1 0 0 b b a a
```

### Description

Add MRb to the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb + #16FHi:0;
```

This instruction can also be written as MADDF32 MRa, MRb, #16FHi.

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

### Pipeline

This is a single-cycle instruction.

### Example

```
; Add to MR1 the value 2.0 in 32-bit floating-point format
; Store the result in MR0
MADDF32 MR0, #2.0, MR1 ; MR0 = 2.0 + MR1

; Add to MR3 the value -2.5 in 32-bit floating-point format
; Store the result in MR2
MADDF32 MR2, #-2.5, MR3 ; MR2 = -2.5 + MR3

; Add to MR3 the value 0x3FC00000 (1.5)
; Store the result in MR3
MADDF32 MR3, #0x3FC0, MR3 ; MR3 = 1.5 + MR3
```

### See also

[MADDF32 MRa, MRb, #16FHi](#)  
[MADDF32 MRa, MRb, MRc](#)  
[MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)  
[MADDF32 MRd, MRe, MRf || MMOV32 mem32, MRa](#)  
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf](#)

**MADDF32 MRa, MRb, #16FHi 32-Bit Floating-Point Addition**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)
#16FHi	A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0.

**Opcode**

```
LSW: I I I I I I I I I I I I I I
MSW: 0 1 1 1 0 1 1 1 1 1 0 0 b b a a
```

**Description**

Add MRb to the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb + #16FHi:0;
```

This instruction can also be written as MADDF32 MRa, #16FHi, MRb.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
; X is an array of 32-bit floating-point values
; Find the maximum value in an array X
; and store it in Result
;
_ClalTask1:
    MMOVI16    MAR1,#_X          ; Start address
    MUI16TOF32 MR0, @_len        ; Length of the array
    MNOF32     MR0, MR0          ; delay for MAR1 load
    MNOF32     MR0, MR0          ; delay for MAR1 load
    MMOV32     MR1, *MAR1[2]++   ; MR1 = X0
LOOP
    MMOV32     MR2, *MAR1[2]++   ; MR2 = next element
    MMAXF32    MR1, MR2          ; MR1 = MAX(MR1, MR2)
    MADDF32    MR0, MR0, #-1.0   ; Decrement the counter
    MCMFP32    MR0 #0.0         ; Set/clear flags for MBCNDD
    MNOF32     MR0, MR0
    MNOF32     MR0, MR0
    MNOF32     MR0, MR0
    MBCNDD    LOOP, NEQ          ; Branch if not equal to zero
    MMOV32    @_Result, MR1      ; Always executed
    MNOF32     MR0, MR0          ; Always executed
    MNOF32     MR0, MR0          ; Always executed
    MSTOP32    MR0, MR0          ; End of task
```



**Example 2**

```

; Show the basic operation of MADDF32
;
; Add to MR1 the value 2.0 in 32-bit floating-point format
; Store the result in MR0
    MADDF32 MR0, MR1, #2.0    ; MR0 = MR1 + 2.0

; Add to MR3 the value -2.5 in 32-bit floating-point format
; Store the result in MR2
    MADDF32 MR2, MR3, #-2.5  ; MR2 = MR3 + (-2.5)

; Add to MR0 the value 0x3FC00000 (1.5)
; Store the result in MR0
    MADDF32 MR0, MR0, #0x3FC0 ; MR0 = MR0 + 1.5

```

**See also**

[MADDF32 MRa, #16FHi, MRb](#)  
[MADDF32 MRa, MRb, MRc](#)  
[MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)  
[MADDF32 MRd, MRe, MRf || MMOV32 mem32, MRa](#)  
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf](#)

**MADDF32 MRa, MRb, MRc 32-Bit Floating-Point Addition**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)
MRc	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 000 0000 00cc bbaa
MSW: 0111 1100 0010 0000
```

**Description**

Add the contents of MRc to the contents of MRb and load the result into MRa.

$$MRa = MRb + MRc;$$
**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given M1, X1 and B1 are 32-bit floating point numbers
; Calculate Y1 = M1*X1+B1
;
_ClalTask1:
    MMOV32 MR0,@M1      ; Load MR0 with M1
    MMOV32 MR1,@X1      ; Load MR1 with X1
    MMPYF32 MR1,MR1,MR0 ; Multiply M1*X1
|| MMOV32 MR0,@B1      ; and in parallel load MR0 with B1
    MADDF32 MR1,MR1,MR0 ; Add M*X1 to B1 and store in MR1
    MMOV32 @Y1,MR1     ; Store the result
    MSTOP              ; end of task
```

**See also**

[MADDF32 MRa, #16FHi, MRb](#)  
[MADDF32 MRa, MRb, #16FHi](#)  
[MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)  
[MADDF32 MRd, MRe, MRf || MMOV32 mem32, MRa](#)  
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf](#)

**MADDF32 MRd, MRe, MRf||MMOV32 mem32, MRa 32-Bit Floating-Point Addition with Parallel Move**
**Operands**

MRd	CLA floating-point destination register for the MADDF32 (MR0 to MR3)
MRe	CLA floating-point source register for the MADDF32 (MR0 to MR3)
MRf	CLA floating-point source register for the MADDF32 (MR0 to MR3)
mem32	32-bit memory location accessed using one of the available addressing modes. This will be the destination of the MMOV32.
MRa	CLA floating-point source register for the MMOV32 (MR0 to MR3)

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0101 ffee ddaa addr
```

**Description**

Perform an MADDF32 and a MMOV32 in parallel. Add MRf to the contents of MRe and store the result in MRd. In parallel move the contents of MRa to the 32-bit location mem32.

```
MRd = MRe + MRf;
[mem32] = MRa;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

**Pipeline**

Both MADDF32 and MMOV32 complete in a single cycle.

**Example**

```
; Given A, B and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A * B) + C
;
;
_Cla1Task2:
    MMOV32  MR0, @_A      ; Load MR0 with A
    MMOV32  MR1, @_B      ; Load MR1 with B
    MMPYF32 MR1, MR1, MR0 ; Multiply A*B
| | MMOV32  MR0, @_C      ; and in parallel load MR0 with C
    MADDF32 MR1, MR1, MR0 ; Add (A*B) to C
| | MMOV32  @_Y2, MR1     ; and in parallel store A*B
    MMOV32  @_Y3, MR1     ; Store the A*B + C
    MSTOP                ; end of task
```

**See also**

[MADDF32 MRa, #16FHi, MRb](#)  
[MADDF32 MRa, MRb, #16FHi](#)  
[MADDF32 MRa, MRb, MRc](#)  
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf](#)  
[MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)

**MADDF32 MRd, MRe, MRf ||MMOV32 MRa, mem32 32-Bit Floating-Point Addition with Parallel Move**
**Operands**

MRd	CLA floating-point destination register for the MADDF32 (MR0 to MR3). MRd cannot be the same register as MRa.
MRe	CLA floating-point source register for the MADDF32 (MR0 to MR3)
MRf	CLA floating-point source register for the MADDF32 (MR0 to MR3)
MRa	CLA floating-point destination register for the MMOV32 (MR0 to MR3). MRa cannot be the same register as MRd.
mem32	32-bit memory location accessed using one of the available addressing modes. This is the source for the MMOV32.

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0001 ffee ddaa addr
```

**Description**

Perform an MADDF32 and a MMOV32 operation in parallel. Add MRf to the contents of MRe and store the result in MRd. In parallel move the contents of the 32-bit location mem32 to MRa.

```
MRd = MRe + MRf;
MRa = [mem32];
```

**Restrictions**

The destination register for the MADDF32 and the MMOV32 must be unique. That is, MRa and MRd cannot be the same register.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

The MMOV32 Instruction will set the NF and ZF flags as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; };
```

**Pipeline**

The MADDF32 and the MMOV32 both complete in a single cycle.

**Example 1**

```
; Given A, B and C are 32-bit floating-point numbers
; Calculate Y1 = A + 4B
;           Y2 = A + C
;
_ClalTask1:
    MMOV32 MR0, @A           ; Load MR0 with A
    MMOV32 MR1, @B           ; Load MR1 with B
    MMPYF32 MR1, MR1, #4.0 ; Multiply 4 * B
||  MMOV32 MR2, @C           ; and in parallel load C
    MADDF32 MR3, MR0, MR1   ; Add A + 4B
    MADDF32 MR3, MR0, MR2   ; Add A + C
||  MMOV32 @Y1, MR3         ; and in parallel store A+4B
    MMOV32 @Y2, MR3         ; store A + C MSTOP
; end of task
```

**Example 2**

```

; Given A, B and C are 32-bit floating-point numbers
; Calculate Y3 = (A + B)
;           Y4 = (A + B) * C
;
_ClalTask2:
    MMOV32 MR0, @A           ; Load MR0 with A
    MMOV32 MR1, @B           ; Load MR1 with B
    MADDF32 MR1, MR1, MR0    ; Add A+B
||  MMOV32 MR0, @C           ; and in parallel load MR0 with C
    MMPYF32 MR1, MR1, MR0    ; Multiply (A+B) by C
||  MMOV32 @Y3, MR1          ; and in parallel store A+B
    MMOV32 @Y4, MR1          ; Store the (A+B) * C
    MSTOP                    ; end of task

```

**See also**

[MADDF32 MRa, #16FHi, MRb](#)  
[MADDF32 MRa, MRb, #16FHi](#)  
[MADDF32 MRa, MRb, MRc](#)  
[MADDF32 MRd, MRe, MRf || MMOV32 mem32, MRa](#)  
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf](#)

**MAND32 MRa, MRb, MRc Bitwise AND**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)
MRc	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 0110 0000
```

**Description**

Bitwise AND of MRb with MRc.

```
MRa(31:0) = MRb(31:0) AND MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ MR0, #0x5555 ; MR0 = 0x5555AAAA
MMOVXI MR0, #0xAAAA

MMOVIZ MR1, #0x5432 ; MR1 = 0x5432FEDC
MMOVXI MR1, #0xFEDC

; 0101 AND 0101 = 0101 (5)
; 0101 AND 0100 = 0100 (4)
; 0101 AND 0011 = 0001 (1)
; 0101 AND 0010 = 0000 (0)
; 1010 AND 1111 = 1010 (A)
; 1010 AND 1110 = 1010 (A)
; 1010 AND 1101 = 1000 (8)
; 1010 AND 1100 = 1000 (8)

MAND32 MR2, MR1, MR0 ; MR3 = 0x5410AA88
```

**See also**

[MADD32 MRa, MRb, MRc](#)  
[MASR32 MRa, #SHIFT](#)  
[MLSL32 MRa, #SHIFT](#)  
[MLSR32 MRa, #SHIFT](#)  
[MOR32 MRa, MRb, MRc](#)  
[MXOR32 MRa, MRb, MRc](#)  
[MSUB32 MRa, MRb, MRc](#)

## MASR32 MRa, #SHIFT *Arithmetic Shift Right*

### Operands

MRa	CLA floating-point source/destination register (MR0 to MR3)
#SHIFT	Number of bits to shift (1 to 32)

### Opcode

```
LSW: 0000 0000 0shi ftaa
MSW: 0111 1011 0100 0000
```

### Description

Arithmetic shift right of MRa by the number of bits indicated. The number of bits can be 1 to 32.

```
MARa(31:0) = Arithmetic Shift(MARa(31:0) by #SHIFT bits);
```

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

### Pipeline

This is a single-cycle instruction.

### Example

```
; Given m2 = (int32)32
;         x2 = (int32)64
;         b2 = (int32)-128
;
; Calculate
;         m2 = m2/2
;         x2 = x2/4
;         b2 = b2/8
;
_ClalTask2:
    MMOV32 MR0, @_m2 ; MR0 = 32 (0x00000020)
    MMOV32 MR1, @_x2 ; MR1 = 64 (0x00000040)
    MMOV32 MR2, @_b2 ; MR2 = -128 (0xFFFFFFFF80)
    MASR32 MR0, #1 ; MR0 = 16 (0x00000010)
    MASR32 MR1, #2 ; MR1 = 16 (0x00000010)
    MASR32 MR2, #3 ; MR2 = -16 (0xFFFFFFFFF0)
    MMOV32 @_m2, MR0 ; store results
    MMOV32 @_x2, MR1
    MMOV32 @_b2, MR2
    MSTOP ; end of task
```

### See also

[MADD32 MRa, MRb, MRc](#)  
[MAND32 MRa, MRb, MRc](#)  
[MLSL32 MRa, #SHIFT](#)  
[MLSR32 MRa, #SHIFT](#)  
[MOR32 MRa, MRb, MRc](#)  
[MXOR32 MRa, MRb, MRc](#)  
[MSUB32 MRa, MRb, MRc](#)

## MBCNDD 16BitDest {, CNDF} *Branch Conditional Delayed*

### Operands

16BitDest	16-bit destination if condition is true
CNDF	Optional condition tested

### Opcode

LSW: dest dest dest dest  
MSW: 0111 1001 1000 cndf

### Description

If the specified condition is true, then branch by adding the signed 16BitDest value to the MPC value. Otherwise, continue without branching. If the address overflows, it wraps around. Therefore a value of "0xFFFFE" will put the MPC back to the MBCNDD instruction.

Please refer to the pipeline section for important information regarding this instruction.

```
if (CNDF == TRUE) MPC += 16BitDest;
```

CNDF is one of the following conditions:

Encode <sup>(1)</sup>	CNDF	Description	MSTF Flags Tested
0000	NEQ	Not equal to zero	ZF == 0
0001	EQ	Equal to zero	ZF == 1
0010	GT	Greater than zero	ZF == 0 AND NF == 0
0011	GEQ	Greater than or equal to zero	NF == 0
0100	LT	Less than zero	NF == 1
0101	LEQ	Less than or equal to zero	ZF == 1 OR NF == 1
1010	TF	Test flag set	TF == 1
1011	NTF	Test flag not set	TF == 0
1100	LU	Latched underflow	LUF == 1
1101	LV	Latched overflow	LVF == 1
1110	UNC	Unconditional	None
1111	UNCF <sup>(2)</sup>	Unconditional with flag modification	None

<sup>(1)</sup> Values not shown are reserved.

<sup>(2)</sup> This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

### Restrictions

The MBCNDD instruction is not allowed three instructions before or after a MBCNDD, MCCNDD or MRCNDD instruction. Refer to the pipeline section for more information.

### Flags

This instruction does not modify flags in the MSTF register.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

### Pipeline

The MBCNDD instruction by itself is a single-cycle instruction. As shown in [Table 5-13](#) for each branch 6 instruction slots are executed; three before the branch instruction (12-14) and three after the branch instruction (15-17). The total number of cycles for a branch taken or not taken depends on the usage of these slots. That is, the number of cycles depends on how many slots are filled with a MNOP as well as which slots are filled. The effective number of cycles for a branch can, therefore, range from 1 to 7 cycles. The number of cycles for a branch taken may not be the same as for a branch not taken.



Referring to [Table 5-13](#) and [Table 5-14](#), the instructions before and after MBCNDD have the following properties:

- **I1**
  - I1 is the last instruction that can effect the CNDF flags for the MBCNDD instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to branch or not when MBCNDD is in the D2 phase.
  - There are no restrictions on the type of instruction for I1.
- **I2, I3 and I4**
  - The three instructions proceeding MBCNDD can change MSTF flags but will have no effect on whether the MBCNDD instruction branches or not. This is because the flag modification will occur after the D2 phase of the MBCNDD instruction.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.
- **I5, I6 and I7**
  - The three instructions following MBCNDD are always executed irrespective of whether the branch is taken or not.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.

```

<Instruction 1> ; I1 Last instruction that can affect flags for
                ; the MBCNDD operation
<Instruction 2> ; I2 Cannot be stop, branch, call or return
<Instruction 3> ; I3 Cannot be stop, branch, call or return
<Instruction 4> ; I4 Cannot be stop, branch, call or return
MBCNDD _Skip, NEQ ; Branch to Skip if not equal to zero
                ; Three instructions after MBCNDD are always
                ; executed whether the branch is taken or not
<Instruction 5> ; I5 Cannot be stop, branch, call or return
<Instruction 6> ; I6 Cannot be stop, branch, call or return
<Instruction 7> ; I7 Cannot be stop, branch, call or return
<Instruction 8> ; I8
<Instruction 9> ; I9
.....
_Skip:
<Destination 1> ; d1 Can be any instruction
<Destination 2> ; d2
<Destination 3> ; d3
.....
.....
MSTOP
.....

```

**Table 5-13. Pipeline Activity For MBCNDD, Branch Not Taken**

Instruction	F1	F2	D1	D2	R1	R2	E	W
I1	I1							
I2	I2	I1						
I3	I3	I2	I1					
I4	I4	I3	I2	I1				
MBCNDD	MBCNDD	I4	I3	I2	I1			
I5	I5	MBCNDD	I4	I3	I2	I1		
I6	I6	I5	MBCNDD	I4	I3	I2	I1	
I7	I7	I6	I5	MBCNDD	I4	I3	I2	
I8	I8	I7	I6	I5	-	I4	I3	
I9	I9	I8	I7	I6	I5	-	I4	
I10	I10	I9	I8	I7	I6	I5	-	
		I10	I9	I8	I7	I6	I5	
			I10	I9	I8	I7	I6	
				I10	I9	I8	I7	
					I10	I9	I8	
						I10	I9	
							I10	

**Table 5-14. Pipeline Activity For MBCNDD, Branch Taken**

Instruction	F1	F2	D1	D2	R1	R2	E	W
I1	I1							
I2	I2	I1						
I3	I3	I2	I1					
I4	I4	I3	I2	I1				
MBCNDD	MBCNDD	I4	I3	I2	I1			
I5	I5	MBCNDD	I4	I3	I2	I1		
I6	I6	I5	MBCNDD	I4	I3	I2	I1	
I7	I7	I6	I5	MBCNDD	I4	I3	I2	
d1	d1	I7	I6	I5	-	I4	I3	
d2	d2	d1	I7	I6	I5	-	I4	
d3	d3	d2	d1	I7	I6	I5	-	
		d3	d2	d1	I7	I6	I5	
			d3	d2	d1	I7	I6	
				d3	d2	d1	I7	
					d3	d2	d1	
						d3	d2	
							d3	

**Example 1**

```

; if (State == 0.1)
; RampState = RampState || RAMPMASK
; else if (State == 0.01)
; CoastState = CoastState || COASTMASK
; else
; SteadyState = SteadyState || STEADYMASK
;
_ClalTask1:
MMOV32 MR0, @State
MCMPPF32 MR0, #0.1          ; Affects flags for 1st MBCNDD (A)
MNOP
MNOP
MNOP
MBCNDD Skip1, NEQ          ; (A) If State != 0.1, go to Skip1
MNOP ; Always executed
MNOP ; Always executed
MNOP ; Always executed
MMOV32 MR1, @RampState      ; Execute if (A) branch not taken
MMOVXI MR2, #RAMPMASK      ; Execute if (A) branch not taken
MOR32 MR1, MR2             ; Execute if (A) branch not taken
MMOV32 @RampState, MR1     ; Execute if (A) branch not taken
MSTOP                      ; end of task if (A) branch not taken
Skip1:
MCMPPF32 MR0,#0.01        ; Affects flags for 2nd MBCNDD (B)
MNOP
MNOP
MNOP
MBCNDD Skip2,NEQ          ; (B) If State != 0.01, go to Skip2
MNOP ; Always executed
MNOP ; Always executed
MNOP ; Always executed
MMOV32 MR1, @CoastState    ; Execute if (B) branch not taken
MMOVXI MR2, #COASTMASK    ; Execute if (B) branch not taken
MOR32 MR1, MR2            ; Execute if (B) branch not taken
MMOV32 @CoastState, MR1   ; Execute if (B) branch not taken
MSTOP
Skip2:
MMOV32 MR3, @SteadyState   ; Executed if (B) branch taken
MMOVXI MR2, #STEADYMASK   ; Executed if (B) branch taken
MOR32 MR3, MR2            ; Executed if (B) branch taken
MMOV32 @SteadyState, MR3  ; Executed if (B) branch taken
MSTOP

```

**Example 2**

```

; This example is the same as Example 1, except
; the code is optimized to take advantage of delay slots
;
; if (State == 0.1)
; RampState = RampState || RAMPMASK
; else if (State == 0.01)
; CoastState = CoastState || COASTMASK
; else
; SteadyState = SteadyState || STEADYMASK
;
_ClalTask2:
MMOV32 MR0, @State
MCMPF32 MR0, #0.1           ; Affects flags for 1st MBCNDD (A)
MCMPF32 MR0, #0.01         ; Check used by 2nd MBCNDD (B)
MTESTTF EQ                 ; Store EQ flag in TF for 2nd MBCNDD (B)
MNOP
MBCNDD Skip1, NEQ          ; (A) If State != 0.1, go to Skip1
MMOV32 MR1, @RampState     ; Always executed
MMOVXI MR2, #RAMPMASK     ; Always executed
MOR32 MR1, MR2             ; Always executed
MMOV32 @RampState, MR1     ; Execute if (A) branch not taken
MSTOP                     ; end of task if (A) branch not taken

Skip1:
MMOV32 MR3, @SteadyState
MMOVXI MR2, #STEADYMASK
MOR32 MR3, MR2
MBCNDD Skip2, NTF         ; (B) if State != .01, go to Skip2
MMOV32 MR1, @CoastState   ; Always executed
MMOVXI MR2, #COASTMASK    ; Always executed
MOR32 MR1, MR2           ; Always executed
MMOV32 @CoastState, MR1   ; Execute if (B) branch not taken
MSTOP                     ; end of task if (B) branch not taken

Skip2:
MMOV32 @SteadyState, MR3  ; Executed if (B) branch taken
MSTOP

```

**See also**

[MCCNDD 16BitDest, CNDF](#)  
[MRCNDD CNDF](#)

## MCCNDD 16BitDest {, CNDF} *Call Conditional Delayed*

### Operands

16BitDest	16-bit destination if condition is true
CNDF	Optional condition to be tested

### Opcode

LSW: dest dest dest dest  
MSW: 0111 1001 1001 cndf

### Description

If the specified condition is true, then store the return address in the RPC field of MSTF and make the call by adding the signed 16BitDest value to the MPC value. Otherwise, continue code execution without making the call. If the address overflows, it wraps around. Therefore a value of "0xFFFFE" will put the MPC back to the MCCNDD instruction.

Please refer to the pipeline section for important information regarding this instruction.

```
if (CNDF == TRUE)
{
    RPC = return address;
    MPC += 16BitDest;
};
```

CNDF is one of the following conditions:

Encode <sup>(3)</sup>	CNDF	Description	MSTF Flags Tested
0000	NEQ	Not equal to zero	ZF == 0
0001	EQ	Equal to zero	ZF == 1
0010	GT	Greater than zero	ZF == 0 AND NF == 0
0011	GEQ	Greater than or equal to zero	NF == 0
0100	LT	Less than zero	NF == 1
0101	LEQ	Less than or equal to zero	ZF == 1 OR NF == 1
1010	TF	Test flag set	TF == 1
1011	NTF	Test flag not set	TF == 0
1100	LU	Latched underflow	LUF == 1
1101	LV	Latched overflow	LVF == 1
1110	UNC	Unconditional	None
1111	UNCF <sup>(4)</sup>	Unconditional with flag modification	None

<sup>(3)</sup> Values not shown are reserved.

<sup>(4)</sup> This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

### Restrictions

The MCCNDD instruction is not allowed three instructions before or after a MBCNDD, MCCNDD, or MRCNDD instruction. Refer to the Pipeline section for more details.

### Flags

This instruction does not modify flags in the MSTF register.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

The MCCNDD instruction by itself is a single-cycle instruction. As shown in [Table 5-15](#), for each call 6 instruction slots are executed; three before the call instruction (I2-I4) and three after the call instruction (I5-I7). The total number of cycles for a call taken or not taken depends on the usage of these slots. That is, the number of cycles depends on how many slots are filled with a MNOP as well as which slots are filled. The effective number of cycles for a call can, therefore, range from 1 to 7 cycles. The number of cycles for a call taken may not be the same as for a call not taken.

Referring to the following code fragment and the pipeline diagrams in [Table 5-15](#) and [Table 5-16](#), the instructions before and after MCCNDD have the following properties:

- **I1**
  - I1 is the last instruction that can effect the CNDF flags for the MCCNDD instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to branch or not when MCCNDD is in the D2 phase.
  - There are no restrictions on the type of instruction for I1.
- **I2, I3 and I4**
  - The three instructions preceding MCCNDD can change MSTF flags but will have no effect on whether the MCCNDD instruction makes the call or not. This is because the flag modification will occur after the D2 phase of the MCCNDD instruction.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.
- **I5, I6 and I7**
  - The three instructions following MBCNDD are always executed irrespective of whether the branch is taken or not.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.

```

<Instruction 1> ; I1 Last instruction that can affect flags for
                ; the MCCNDD operation
<Instruction 2> ; I2 Cannot be stop, branch, call or return
<Instruction 3> ; I3 Cannot be stop, branch, call or return
<Instruction 4> ; I4 Cannot be stop, branch, call or return

MCCNDD _func, NEQ ; Call to func if not equal to zero

                ; Three instructions after MCCNDD are always
                ; executed whether the call is taken or not

<Instruction 5> ; I5 Cannot be stop, branch, call or return
<Instruction 6> ; I6 Cannot be stop, branch, call or return
<Instruction 7> ; I7 Cannot be stop, branch, call or return
<Instruction 8> ; I8 The address of this instruction is saved
                ; in the RPC field of the MSTF register.
                ; Upon return this value is loaded into MPC
                ; and fetching continues from this point.
<Instruction 9> ; I9
....
_func:
<Destination 1> ; d1 Can be any instruction
<Destination 2> ; d2
<Destination 3> ; d3
<Destination 4> ; d4 Last instruction that can affect flags for
                ; the MRCNDD operation

<Destination 5> ; d5 Cannot be stop, branch, call or return
<Destination 6> ; d6 Cannot be stop, branch, call or return
<Destination 7> ; d7 Cannot be stop, branch, call or return

MRCNDD UNC      ; Return to <Instruction 8>, unconditional

                ; Three instructions after MRCNDD are always
                ; executed whether the return is taken or not

<Destination 8> ; d8 Cannot be stop, branch, call or return
<Destination 9> ; d9 Cannot be stop, branch, call or return
<Destination 10> ; d10 Cannot be stop, branch, call or return
<Destination 11> ; d11
....
MSTOP

```

**Table 5-15. Pipeline Activity For MCCNDD, Call Not Taken**

Instruction	F1	F2	D1	D2	R1	R2	E	W
I1	I1							
I2	I2	I1						
I3	I3	I2	I1					
I4	I4	I3	I2	I1				
MCCNDD	MCCNDD	I4	I3	I2	I1			
I5	I5	MCCNDD	I4	I3	I2	I1		
I6	I6	I5	MCCNDD	I4	I3	I2	I1	
I7	I7	I6	I5	MCCNDD	I4	I3	I2	
I8	I8	I7	I6	I5	-	I4	I3	
I9	I9	I8	I7	I6	I5	-	I4	
I10	I10	I9	I8	I7	I6	I5	-	
etc ....		I10	I9	I8	I7	I6	I5	
....			I10	I9	I8	I7	I6	
....				I10	I9	I8	I7	
....					I10	I9	I8	
						I10	I9	
							I10	

**Table 5-16. Pipeline Activity For MCCNDD, Call Taken**

Instruction	F1	F2	D1	D2	R1	R2	E	W
I1	I1							
I2	I2	I1						
I3	I3	I2	I1					
I4	I4	I3	I2	I1				
MCCNDD	MCCNDD	I4	I3	I2	I1			
I5	I5	MCCNDD	I4	I3	I2	I1		
I6	I6	I5	MCCNDD	I4	I3	I2	I1	
I7 <sup>(1)</sup>	I7	I6	I5	MCCNDD	I4	I3	I2	
d1	d1	I7	I6	I5	-	I4	I3	
d2	d2	d1	I7	I6	I5	-	I4	
d3	d3	d2	d1	I7	I6	I5	-	
etc ....		d3	d2	d1	I7	I6	I5	
....			d3	d2	d1	I7	I6	
....				d3	d2	d1	I7	
....					d3	d2	d1	
						d3	d2	
							d3	

<sup>(1)</sup> The RPC value in the MSTF register will point to the instruction following I7 (instruction I8).

**Example**

;

**See also**

[MBCNDD #16BitDest, CNDF](#)  
[MMOV32 mem32, MSTF](#)  
[MMOV32 MSTF, mem32](#)  
[MRCNDD CNDF](#)



**MCLRC BGINTM**      *Clear Background Task Interrupt Mask*
**Operands**

None                      This instruction does not have any operands

**Opcode**

LSW: 0000 0000 0000 0000  
MSW: 0111 1111 0111 0000

**Description**

This instruction will clear the background task interrupt mask (BGINTM) bit in the MSTSBGRND register, allowing any code thereafter to be interrupted by a higher priority task. This instruction clears the BGINTM bit at the end of its D2 phase.

Note: This instruction does not require the MEALLOW bit to be asserted before or deasserted after clearing BGINTM.

**Flags**

This instruction does not modify flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MCLRC BGINTM      ; Allow the background task to be
                   ; interrupted by clearing the
                   ; MSTSBGRND.BGINTM bit
```

**See also**

[MSETC BGINTM](#)

**MCMP32 MRa, MRb 32-Bit Integer Compare for Equal, Less Than or Greater Than**
**Operands**

MRa	CLA floating-point source register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1111 0010 0000
```

**Description**

Set ZF and NF flags on the result of MRa - MRb where MRa and MRb are 32-bit integers. For a floating point compare refer to [MCMFP32](#).

**NOTE:** A known hardware issue exists in the MCMP32 instruction. Signed integer comparisons using MCMP32 by itself will set the status bits in a way that is not useful for comparison when the difference between the two operands is too large, such as when the inputs have opposite sign and are near the extreme 32-bit signed values. This affects both signed and unsigned integer comparisons.

The compiler (version 18.1.5.LTS or higher) has implemented a workaround for this issue. The compiler can check the upper bits of the operands by performing a floating point comparison before proceeding to do the integer comparison or subtraction.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified based on the integer results of the operation.

```
If(MRa == MRb) {ZF=1; NF=0;}
If(MRa > MRb) {ZF=0; NF=0;}
If(MRa < MRb) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Behavior of ZF and NF flags for different comparisons
;
; Given A = (int32)1
;       B = (int32)2
;       C = (int32)-7
;
MMOV32 MR0, @_A ; MR0 = 1 (0x00000001)
MMOV32 MR1, @_B ; MR1 = 2 (0x00000002)
MMOV32 MR2, @_C ; MR2 = -7 (0xFFFFFFFF9)
MCMP32 MR2, MR2 ; NF = 0, ZF = 1
MCMP32 MR0, MR1 ; NF = 1, ZF = 0
MCMP32 MR1, MR0 ; NF = 0, ZF = 0
```

**See also**

[MADD32 MRa, MRb, MRc](#)  
[MSUB32 MRa, MRb, MRc](#)

## MCMPF32 MRa, MRb 32-Bit Floating-Point Compare for Equal, Less Than or Greater Than

### Operands

MRa	CLA floating-point source register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

### Opcode

LSW: 0000 0000 0000 bbaa  
MSW: 0111 1101 0000 0000

### Description

Set ZF and NF flags on the result of MRa - MRb. The MCMPF32 instruction is performed as a logical compare operation. This is possible because of the IEEE format offsetting the exponent. Basically the bigger the binary number, the bigger the floating-point value.

Special cases for inputs:

- Negative zero will be treated as positive zero.
- A denormalized value will be treated as positive zero.
- Not-a-Number (NaN) will be treated as infinity.

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified as follows:

```
If(MRa == MRb) {ZF=1; NF=0;}
If(MRa > MRb) {ZF=0; NF=0;}
If(MRa < MRb) {ZF=0; NF=1;}
```

### Pipeline

This is a single-cycle instruction.

### Example

; Behavior of ZF and NF flags for different comparisons

```
MMOVIZ MR1, #-2.0 ; MR1 = -2.0 (0xC0000000)
MMOVIZ MR0, #5.0 ; MR0 = 5.0 (0x40A00000)
MCMPF32 MR1, MR0 ; ZF = 0, NF = 1
MCMPF32 MR0, MR1 ; ZF = 0, NF = 0
MCMPF32 MR0, MR0 ; ZF = 1, NF = 0
```

### See also

[MCMPF32 MRa, #16FHi](#)  
[MMAXF32 MRa, #16FHi](#)  
[MMAXF32 MRa, MRb](#)  
[MMINF32 MRa, #16FHi](#)  
[MMINF32 MRa, MRb](#)

**MCMPF32 MRa, #16FHi 32-Bit Floating-Point Compare for Equal, Less Than or Greater Than**
**Operands**

MRa	CLA floating-point source register (MR0 to MR3)
#16FHi	A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0.

**Opcode**

```
LSW: I I I I I I I I I I I I I I
MSW: 0 1 1 1 1 0 0 0 1 1 0 0 0 0 a a
```

**Description**

Compare the value in MRa with the floating-point value represented by the immediate operand. Set the ZF and NF flags on (MRa - #16FHi:0).

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. This addressing mode is most useful for constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

The MCMPF32 instruction is performed as a logical compare operation. This is possible because of the IEEE floating-point format offsets the exponent. Basically the bigger the binary number, the bigger the floating-point value.

Special cases for inputs:

- Negative zero will be treated as positive zero.
- Denormalized value will be treated as positive zero.
- Not-a-Number (NaN) will be treated as infinity.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified as follows:

```
If (MRa == #16FHi:0) {ZF=1, NF=0;}
If (MRa > #16FHi:0) {ZF=0, NF=0;}
If (MRa < #16FHi:0) {ZF=0, NF=1;}
```

**Pipeline**

This is a single-cycle instruction

**Example 1**

; Behavior of ZF and NF flags for different comparisons

```
MMOVIZ MR1, #-2.0 ; MR1 = -2.0 (0xC0000000)
MMOVIZ MR0, #5.0 ; MR0 = 5.0 (0x40A00000)
MCMPF32 MR1, #-2.2 ; ZF = 0, NF = 0
MCMPF32 MR0, #6.5 ; ZF = 0, NF = 1
MCMPF32 MR0, #5.0 ; ZF = 1, NF = 0
```

**Example 2**

```

; X is an array of 32-bit floating-point values
; and has len elements. Find the maximum value in
; the array and store it in Result
;
; Note: MCMPPF32 and MSWAPF can be replaced with MMAXF32
;
_ClalTask1:
  MMOV16 MAR1,#_X      ; Start address
  MUI16TOF32 MR0, @_len ; Length of the array
  MNOP                 ; delay for MAR1 load
  MNOP                 ; delay for MAR1 load
  MMOV32 MR1, *MAR1[2]++ ; MR1 = X0

LOOP
  MMOV32 MR2, *MAR1[2]++ ; MR2 = next element
  MCMPPF32 MR2, MR1      ; Compare MR2 with MR1
  MSWAPF MR1, MR2, GT   ; MR1 = MAX(MR1, MR2)
  MADD32 MR0, MR0, #-1.0 ; Decrement the counter
  MCMPPF32 MR0 #0.0     ; Set/clear flags for MBCNDD
  MNOP
  MNOP
  MNOP
  MBCNDD LOOP, NEQ     ; Branch if not equal to zero
  MMOV32 @_Result, MR1 ; Always executed
  MNOP                 ; Always executed
  MNOP                 ; Always executed
  MSTOP                ; End of task

```

**See also**

[MCMPPF32 MRa, MRb](#)  
[MMAXF32 MRa, #16FHi](#)  
[MMAXF32 MRa, MRb](#)  
[MMINF32 MRa, #16FHi](#)  
[MMINF32 MRa, MRb](#)

**MDEBUGSTOP**
***Debug Stop Task***
**Operands**

none                      This instruction does not have any operands

**Opcode**

LSW: 0000 0000 0000 0000  
MSW: 0111 1111 0110 0000

**Description**

When CLA breakpoints are enabled, the MDEBUGSTOP instruction is used to halt a task so that it can be debugged. That is, MDEBUGSTOP is the CLA breakpoint. If CLA breakpoints are not enabled, the MDEBUGSTOP instruction behaves like a MNOP. Unlike the MSTOP, the MIRUN flag is not cleared and an interrupt is not issued. A single-step or run operation will continue execution of the task.

**Restrictions**

The MDEBUGSTOP instruction cannot be placed 3 instructions before or after a [MBCNDD](#), [MCCNDD](#) or [MRCNDD](#) instruction.

**Flags**

This instruction does not modify flags in the MSTF register.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

;

**See also**

[MSTOP](#), [MDEBUGSTOP1](#)

**MDEBUGSTOP1**      *Software Breakpoint*


---

**Operands**

none                      This instruction does not have any operands

---

**Opcode**

LSW: 0000 0000 0000 0000  
MSW: 0111 1111 0011 0000

**Description**

The instruction at which a software breakpoint is placed will be replaced by the MDEBUGSTOP1 instruction. It will halt execution once it reaches the D2 phase in the pipeline; at that point the subsequent instructions that were fetched, after the halt, will be flushed from the pipeline. The replace instruction will be re-fetched after this and execution can continue normally (either in run or step mode).

See [Section 5.5.3](#) for a detailed explanation of its operation.

**Restrictions**

The MDEBUGSTOP1 instruction cannot be placed 3 instructions before or after a [MBCNDD](#), [MCCNDD](#) or [MRCNDD](#) instruction.

**Flags**

This instruction does not modify flags in the MSTF register.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

---

**Pipeline**

This is a single-cycle instruction.

**Example**

;

**See also**

[MSTOP](#), [MDEBUGSTOP](#)

---

**MEALLOW**                      ***Enable CLA Write Access to EALLOW Protected Registers***


---

**Operands**

none	This instruction does not have any operands
------	---------------------------------------------

---

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1001 0000
```

**Description**

This instruction sets the MEALLOW bit in the CLA status register MSTF. When this bit is set, the CLA is allowed write access to EALLOW protected registers. To again protect against CLA writes to protected registers, use the MEDIS instruction.

MEALLOW and MEDIS only control CLA write access; reads are allowed even if MEALLOW has not been executed. MEALLOW and MEDIS are also independent from the main CPU's EALLOW/EDIS. This instruction does not modify the EALLOW bit in the main CPU's status register. The MEALLOW bit in MSTF only controls access for the CLA while the EALLOW bit in the ST1 register only controls access for the main CPU.

As with EALLOW, the MEALLOW bit is overridden via the JTAG port, allowing full control of register accesses during debug from Code Composer Studio.

**Flags**

This instruction does not modify flags in the MSTF register.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

---

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; C header file including definition of
; the EPwm1Regs structure
;
; The ePWM TZSEL register is EALLOW protected
;
    .cdecls C,LIST, "CLAShared.h"
    ...
_ClalTask1:
    ...
    MEALLOW                ; Allow CLA write access
    MMOV16 @_EPwm1Regs.TZSEL.all, MR3 ; Write to TZSEL
    MEDIS                  ; Disallow CLA write access
    ...
    ...
    MSTOP
```

**See also**

[MEDIS](#)



## MEDIS *Disable CLA Write Access to EALLOW Protected Registers*

### Operands

none This instruction does not have any operands

### Opcode

LSW: 0000 0000 0000 0000  
MSW: 0111 1111 1011 0000

### Description

This instruction clears the MEALLOW bit in the CLA status register MSTF. When this bit is clear, the CLA is not allowed write access to EALLOW-protected registers. To enable CLA writes to protected registers, use the MEALLOW instruction.

MEALLOW and MEDIS only control CLA write access; reads are allowed even if MEALLOW has not been executed. MEALLOW and MEDIS are also independent from the main CPU's EALLOW/EDIS. This instruction does not modify the EALLOW bit in the main CPU's status register. The MEALLOW bit in MSTF only controls access for the CLA while the EALLOW bit in the ST1 register only controls access for the main CPU.

As with EALLOW, the MEALLOW bit is overridden via the JTAG port, allowing full control of register accesses during debug from Code Composer Studio.

### Flags

This instruction does not modify flags in the MSTF register.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

### Pipeline

This is a single-cycle instruction.

### Example

```
; C header file including definition of
; the EPwm1Regs structure
;
; The ePWM TZSEL register is EALLOW protected
;
    .cdecls C,LIST, "CLAShared.h"
    ...
_ClalTask1:
    ...
    MEALLOW                ; Allow CLA write access
    MMOV16 @_EPwm1Regs.TZSEL.all, MR3 ; Write to TZSEL
    MEDIS                  ; Disallow CLA write access
    ...
    ...
    MSTOP
```

### See also

[MEALLOW](#)

**MEINVF32 MRa, MRb 32-Bit Floating-Point Reciprocal Approximation**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1111 0000 0000
```

**Description**

This operation generates an estimate of  $1/X$  in 32-bit floating-point format accurate to approximately 8 bits. This value can be used in a Newton-Raphson algorithm to get a more accurate answer. That is:

```
Ye = Estimate(1/X);
Ye = Ye*(2.0 - Ye*X);
Ye = Ye*(2.0 - Ye*X);
```

After two iterations of the Newton-Raphson algorithm, you will get an exact answer accurate to the 32-bit floating-point format. On each iteration the mantissa bit accuracy approximately doubles. The MEINVF32 operation will not generate a negative zero, DeNorm or NaN value.

```
MRa = Estimate of 1/MRb;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MEINVF32 generates an underflow condition.
- LVF = 1 if MEINVF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Calculate Num/Den using a Newton-Raphson algorithm for 1/Den
; Ye = Estimate(1/X)
; Ye = Ye*(2.0 - Ye*X)
; Ye = Ye*(2.0 - Ye*X)
;
_ClalTask1:
    MMOV32 MR1, @_Den      ; MR1 = Den
    MEINVF32 MR2, MR1      ; MR2 = Ye = Estimate(1/Den)
    MMPYF32 MR3, MR2, MR1 ; MR3 = Ye*Den
    MSUBF32 MR3, #2.0, MR3 ; MR3 = 2.0 - Ye*Den
    MMPYF32 MR2, MR2, MR3 ; MR2 = Ye = Ye*(2.0 - Ye*Den)
    MMPYF32 MR3, MR2, MR1 ; MR3 = Ye*Den
|| MMOV32 MR0, @_Num      ; MR0 = Num
    MSUBF32 MR3, #2.0, MR3 ; MR3 = 2.0 - Ye*Den
    MMPYF32 MR2, MR2, MR3 ; MR2 = Ye = Ye*(2.0 - Ye*Den)
|| MMOV32 MR1, @_Den      ; Reload Den To Set Sign
    MNEGF32 MR0, MR0, EQ   ; if(Den == 0.0) Change Sign Of Num
    MMPYF32 MR0, MR2, MR0 ; MR0 = Y = Ye*Num
    MMOV32 @_Dest, MR0    ; Store result
    MSTOP                  ; end of task
```

**See also**

[MEISQRTF32 MRa, MRb](#)

## MEISQRTF32 MRa, MRb 32-Bit Floating-Point Square-Root Reciprocal Approximation

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

### Opcode

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 0100 0000
```

### Description

This operation generates an estimate of  $1/\sqrt{X}$  in 32-bit floating-point format accurate to approximately 8 bits. This value can be used in a Newton-Raphson algorithm to get a more accurate answer. That is:

```
Ye = Estimate(1/sqrt(X));
Ye = Ye*(1.5 - Ye*Ye*X/2.0);
Ye = Ye*(1.5 - Ye*Ye*X/2.0);
```

After 2 iterations of the Newton-Raphson algorithm, you will get an exact answer accurate to the 32-bit floating-point format. On each iteration the mantissa bit accuracy approximately doubles. The MEISQRTF32 operation will not generate a negative zero, DeNorm or NaN value.

```
MRa = Estimate of 1/sqrt (MRb);
```

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MEISQRTF32 generates an underflow condition.
- LVF = 1 if MEISQRTF32 generates an overflow condition.

### Pipeline

This is a single-cycle instruction.

### Example

```
; Y = sqrt(X)
; Ye = Estimate(1/sqrt(X));
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Y = X*Ye
;
_ClalTask3:
    MMOV32 MR0, @_x          ; MR0 = X
    MEISQRTF32 MR1, MR0      ; MR1 = Ye = Estimate(1/sqrt(X))
    MMOV32 MR1, @_x, EQ      ; if(X == 0.0) Ye = 0.0
    MMPYF32 MR3, MR0, #0.5    ; MR3 = X*0.5
    MMPYF32 MR2, MR1, MR3    ; MR2 = Ye*X*0.5
    MMPYF32 MR2, MR1, MR2    ; MR2 = Ye*Ye*X*0.5
    MSUBF32 MR2, #1.5, MR2   ; MR2 = 1.5 - Ye*Ye*X*0.5
    MMPYF32 MR1, MR1, MR2    ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    MMPYF32 MR2, MR1, MR3    ; MR2 = Ye*X*0.5
    MMPYF32 MR2, MR1, MR2    ; MR2 = Ye*Ye*X*0.5
    MSUBF32 MR2, #1.5, MR2   ; MR2 = 1.5 - Ye*Ye*X*0.5
    MMPYF32 MR1, MR1, MR2    ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    MMPYF32 MR0, MR1, MR0    ; MR0 = Y = Ye*X
    MMOV32 @_y, MR0         ; Store Y = sqrt(X)
    MSTOP                   ; end of task
```

### See also

[MEINVF32 MRa, MRb](#)

**MF32TOI16 MRa, MRb Convert 32-Bit Floating-Point Value to 16-Bit Integer**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1110 0000
```

**Description**

Convert a 32-bit floating point value in MRb to a 16-bit integer and truncate. The result will be stored in MRa.

```
MRa(15:0) = F32TOI16(MRb);
MRa(31:16) = sign extension of MRa(15);
```

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ    MR0, #5.0    ; MR0      = 5.0 (0x40A00000)
MF32TOI16 MR1, MR0    ; MR1(15:0) = MF32TOI16(MR0) = 0x0005
                ; MR1(31:16) = Sign extension of MR1(15) = 0x0000
MMOVIZ    MR2, #-5.0   ; MR2      = -5.0 (0xC0A00000)
MF32TOI16 MR3, MR2    ; MR3(15:0) = MF32TOI16(MR2) = -5 (0xFFFFB)
                ; MR3(31:16) = Sign extension of MR3(15) = 0xFFFF
```

**See also**

[MF32TOI16R MRa, MRb](#)  
[MF32TOUI16 MRa, MRb](#)  
[MF32TOUI16R MRa, MRb](#)  
[MI16TOF32 MRa, MRb](#)  
[MI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, MRb](#)

## MF32TOI16R MRa, MRb *Convert 32-Bit Floating-Point Value to 16-Bit Integer and Round*

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

### Opcode

LSW: 0000 0000 0000 bbaa  
MSW: 0111 1110 0110 0000

### Description

Convert the 32-bit floating point value in MRb to a 16-bit integer and round to the nearest even value. The result is stored in MRa.

MRa(15:0) = F32TOI16round(MRb);  
MRa(31:16) = sign extension of MRa(15);

### Flags

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

### Pipeline

This is a single-cycle instruction.

### Example

```
MMOVIZ MR0, #0x3FD9 ; MR0(31:16) = 0x3FD9
MMOVXI MR0, #0x999A ; MR0(15:0) = 0x999A
                    ; MR0 = 1.7 (0x3FD9999A)
MF32TOI16R MR1, MR0 ; MR1(15:0) = MF32TOI16round (MR0) = 2 (0x0002)
                    ; MR1(31:16) = Sign extension of MR1(15) = 0x0000
MMOVF32 MR2, #-1.7 ; MR2 = -1.7 (0xBFD9999A)
MF32TOI16R MR3, MR2 ; MR3(15:0) = MF32TOI16round (MR2) = -2 (0xFFFFE)
                    ; MR3(31:16) = Sign extension of MR2(15) = 0xFFFF
```

### See also

[MF32TOI16 MRa, MRb](#)  
[MF32TOUI16 MRa, MRb](#)  
[MF32TOUI16R MRa, MRb](#)  
[MI16TOF32 MRa, MRb](#)  
[MI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, MRb](#)

**MF32TOI32 MRa, MRb Convert 32-Bit Floating-Point Value to 32-Bit Integer**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0110 0000
```

**Description**

Convert the 32-bit floating-point value in MRb to a 32-bit integer value and truncate. Store the result in MRa.

```
MRa = F32TOI32(MRb);
```

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
MMOV32 MR2, #11204005.0 ; MR2 = 11204005.0 (0x4B2AF5A5)
MF32TOI32 MR3, MR2 ; MR3 = MF32TOI32(MR2) = 11204005 (0x00AAF5A5)
MMOV32 MR0, #-11204005.0 ; MR0 = -11204005.0 (0xCB2AF5A5)
MF32TOI32 MR1, MR0 ; MR1 = MF32TOI32(MR0) = -11204005 (0xFF550A5B)
```

**Example 2**

```
; Given X, M and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
; Convert M, X and B from IQ24 to float
;
_ClalTask2:
    MI32TOF32 MR0, @_M ; MR0 = 0x4BC00000
    MI32TOF32 MR1, @_X ; MR1 = 0x4C200000
    MI32TOF32 MR2, @_B ; MR2 = 0xCB000000
    MMPYF32 MR0, MR0, #0x3380 ; M = 1/(1*2^24) * iqm = 1.5 (0x3FC00000)
    MMPYF32 MR1, MR1, #0x3380 ; X = 1/(1*2^24) * iqx = 2.5 (0x40200000)
    MMPYF32 MR2, MR2, #0x3380 ; B = 1/(1*2^24) * iqb = -.5 (0xBF000000)
    MMPYF32 MR3, MR0, MR1 ; M*X
    MADDF32 MR2, MR2, MR3 ; Y=MX+B = 3.25 (0x40500000)

; Convert Y from float32 to IQ24
MMPYF32 MR2, MR2, #0x4B80 ; Y * 1*2^24
MF32TOI32 MR2, MR2 ; IQ24(Y) = 0x03400000
MMOV32 @_Y, MR2 ; store result
MSTOP ; end of task
```

**See also**

[MF32TOUI32 MRa, MRb](#)  
[MI32TOF32 MRa, MRb](#)  
[MI32TOF32 MRa, mem32](#)  
[MUI32TOF32 MRa, MRb](#)  
[MUI32TOF32 MRa, mem32](#)

**MF32TOUI16 MRa, MRb Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer**


---

**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1010 0000
```

**Description**

Convert the 32-bit floating point value in MRb to an unsigned 16-bit integer value and truncate to zero. The result will be stored in MRa. To instead round the integer to the nearest even value use the MF32TOUI16R instruction.

```
MRa(15:0) = F32TOUI16(MRb);
MRa(31:16) = 0x0000;
```

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #9.0      ; MR0 = 9.0 (0x41100000)
MF32TOUI16  MR1, MR0      ; MR1(15:0) = MF32TOUI16(MR0) = 9 (0x0009)
              ; MR1(31:16) = 0x0000
MMOVIZ      MR2, #-9.0     ; MR2 = -9.0 (0xC1100000)
MF32TOUI16  MR3, MR2      ; MR3(15:0) = MF32TOUI16(MR2) = 0 (0x0000)
              ; MR3(31:16) = 0x0000
```

**See also**

[MF32TOI16 MRa, MRb](#)  
[MF32TOUI16 MRa, MRb](#)  
[MF32TOUI16R MRa, MRb](#)  
[MI16TOF32 MRa, MRb](#)  
[MI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, MRb](#)

**MF32TOUI16R MRa, MRb *Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer and Round***
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1100 0000
```

**Description**

Convert the 32-bit floating-point value in MRb to an unsigned 16-bit integer and round to the closest even value. The result will be stored in MRa. To instead truncate the converted value, use the MF32TOUI16 instruction.

```
MRa(15:0) = MF32TOUI16round(MRb);
MRa(31:16) = 0x0000;
```

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #0x412C      ; MR0 = 0x412C
MMOVXI      MR0, #0xCCCD      ; MR0 = 0xCCCD ; MR0 = 10.8 (0x412CCCCD)
MF32TOUI16R MR1, MR0          ; MR1(15:0) = MF32TOUI16round(MR0) = 11 (0x000B)
                                   ; MR1(31:16) = 0x0000
MMOVF32     MR2, #-10.8       ; MR2 = -10.8 (0x0xC12CCCCD)
MF32TOUI16R MR3, MR2          ; MR3(15:0) = MF32TOUI16round(MR2) = 0 (0x0000)
                                   ; MR3(31:16) = 0x0000
```

**See also**

[MF32TOI16 MRa, MRb](#)  
[MF32TOI16R MRa, MRb](#)  
[MF32TOUI16 MRa, MRb](#)  
[MI16TOF32 MRa, MRb](#)  
[MI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, MRb](#)



---

**MF32TOUI32 MRa, MRb** *Convert 32-Bit Floating-Point Value to 32-Bit Unsigned Integer*


---

**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

LSW: 0000 0000 0000 bbaa  
MSW: 0111 1101 1010 0000

**Description**

Convert the 32-bit floating-point value in MRb to an unsigned 32-bit integer and store the result in MRa.

MRa = F32TOUI32(MRb);

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #12.5 ; MR0 = 12.5 (0x41480000)
MF32TOUI32  MR0, MR0   ; MR0 = MF32TOUI32 (MR0) = 12 (0x0000000C)
MMOVIZ      MR1, #-6.5 ; MR1 = -6.5 (0xC0D00000)
MF32TOUI32  MR2, MR1   ; MR2 = MF32TOUI32 (MR1) = 0.0 (0x00000000)
```

**See also**

[MF32TOI32 MRa, MRb](#)  
[MI32TOF32 MRa, MRb](#)  
[MI32TOF32 MRa, mem32](#)  
[MUI32TOF32 MRa, MRb](#)  
[MUI32TOF32 MRa, mem32](#)

**MFRACF32 MRa, MRb** *Fractional Portion of a 32-Bit Floating-Point Value*
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

LSW: 0000 0000 0000 bbaa  
MSW: 0111 1110 0000 0000

**Description**

Returns in MRa the fractional portion of the 32-bit floating-point value in MRb

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ    MR2, #19.625 ; MR2 = 19.625 (0x419D0000)
MFRACF32  MR3, MR2    ; MR3 = MFRACF32(MR2) = 0.625 (0x3F200000)0
```

**See also**

## MI16TOF32 MRa, MRb *Convert 16-Bit Integer to 32-Bit Floating-Point Value*

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

### Opcode

LSW: 0000 0000 0000 bbaa  
MSW: 0111 1110 1000 0000

### Description

Convert the 16-bit signed integer in MRb to a 32-bit floating point value and store the result in MRa.

MRa = MI16TOF32(MRb);

### Flags

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

### Pipeline

This is a single-cycle instruction.

### Example

```
MMOVIZ    MR0, #0x0000    ; MR0(31:16) = 0.0 (0x0000)
MMOVXI    MR0, #0x0004    ; MR0(15:0) = 4.0 (0x0004)
MI16TOF32 MR1, MR0        ; MR1 = MI16TOF32 (MR0) = 4.0 (0x40800000)

MMOVIZ    MR2, #0x0000    ; MR2(31:16) = 0.0 (0x0000)
MMOVXI    MR2, #0xFFFC    ; MR2(15:0) = -4.0 (0xFFFC)
MI16TOF32 MR3, MR2        ; MR3 = MI16TOF32 (MR2) = -4.0 (0xC0800000)
MSTOP
```

### See also

[MF32TOI16 MRa, MRb](#)  
[MF32TOI16R MRa, MRb](#)  
[MF32TOUI16 MRa, MRb](#)  
[MF32TOUI16R MRa, MRb](#)  
[MI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, MRb](#)

**MI16TOF32 MRa, mem16 *Convert 16-Bit Integer to 32-Bit Floating-Point Value***
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
mem16	16-bit source memory location to be converted

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 00aa addr
```

**Description**

Convert the 16-bit signed integer indicated by the mem16 pointer to a 32-bit floating-point value and store the result in MRa.

```
MRa = MI16TOF32[mem16];
```

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction:

**Example**

```
; Assume A = 4 (0x0004)
;           B = -4 (0xFFFC)

MI16TOF32 MR0, @_A ; MR0 = MI16TOF32(A) = 4.0 (0x40800000)
MI16TOF32 MR1, @_B ; MR1 = MI16TOF32(B) = -4.0 (0xC0800000)
```

**See also**

[MF32TOI16 MRa, MRb](#)  
[MF32TOI16R MRa, MRb](#)  
[MF32TOUI16 MRa, MRb](#)  
[MF32TOUI16R MRa, MRb](#)  
[MI16TOF32 MRa, MRb](#)  
[MUI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, MRb](#)

## MI32TOF32 MRa, mem32 *Convert 32-Bit Integer to 32-Bit Floating-Point Value*

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
mem32	32-bit memory source for the MMOV32 operation.

### Opcode

LSW: mmmm mmmm mmmm mmmm  
MSW: 0111 0100 01aa addr

### Description

Convert the 32-bit signed integer indicated by mem32 to a 32-bit floating point value and store the result in MRa.

MRa = MI32TOF32[mem32];

### Flags

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

### Pipeline

This is a single-cycle instruction.

### Example

```
; Given X, M and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
; Convert M, X and B from IQ24 to float
;

_ClalTask3:
    MI32TOF32 MR0, @_M      ; MR0 = 0x4BC00000
    MI32TOF32 MR1, @_X      ; MR1 = 0x4C200000
    MI32TOF32 MR2, @_B      ; MR2 = 0xCB000000
    MMPYF32 MR0, MR0, #0x3380 ; M = 1/(1*2^24) * iqm = 1.5 (0x3FC00000)
    MMPYF32 MR1, MR1, #0x3380 ; X = 1/(1*2^24) * iqx = 2.5 (0x40200000)
    MMPYF32 MR2, MR2, #0x3380 ; B = 1/(1*2^24) * iqb = -.5 (0xBF000000)
    MMPYF32 MR3, MR0, MR1    ; M*X
    MADDF32 MR2, MR2, MR3    ; Y=MX+B = 3.25 (0x40500000)

; Convert Y from float32 to IQ24
    MMPYF32 MR2, MR2, #0x4B80 ; Y * 1*2^24
    MF32TOI32 MR2, MR2        ; IQ24(Y) = 0x03400000
    MMOV32 @_Y, MR2          ; store result
    MSTOP                    ; end of task
```

### See also

[MF32TOI32 MRa, MRb](#)  
[MF32TOUI32 MRa, MRb](#)  
[MI32TOF32 MRa, MRb](#)  
[MUI32TOF32 MRa, MRb](#)  
[MUI32TOF32 MRa, mem32](#)

**MI32TOF32 MRa, MRb Convert 32-Bit Integer to 32-Bit Floating-Point Value**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1000 0000
```

**Description**

Convert the signed 32-bit integer in MRb to a 32-bit floating-point value and store the result in MRa.

```
MRa = MI32TOF32(MRb);
```

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Example1:
;
MMOVIZ MR2, #0x1111 ; MR2(31:16) = 4369 (0x1111)
MMOVXI MR2, #0x1111 ; MR2(15:0) = 4369 (0x1111)
; MR2 = +286331153 (0x11111111)
MI32TOF32 MR3, MR2 ; MR3 = MI32TOF32 (MR2) = 286331153.0 (0x4D888888)
```

**See also**

[MF32TOI32 MRa, MRb](#)  
[MF32TOUI32 MRa, MRb](#)  
[MI32TOF32 MRa, mem32](#)  
[MUI32TOF32 MRa, MRb](#)  
[MUI32TOF32 MRa, mem32](#)

## MLSL32 MRa, #SHIFT *Logical Shift Left*

### Operands

MRa	CLA floating-point source/destination register (MR0 to MR3)
#SHIFT	Number of bits to shift (1 to 32)

### Opcode

```
LSW: 0000 0000 0shi ftaa
MSW: 0111 1011 1100 0000
```

### Description

Logical shift left of MRa by the number of bits indicated. The number of bits can be 1 to 32.

```
MARa(31:0) = Logical Shift Left(MARa(31:0) by #SHIFT bits);
```

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

### Pipeline

This is a single-cycle instruction.

### Example

```
; Given m2 = (int32)32
;         x2 = (int32)64
;         b2 = (int32)-128
;
; Calculate:
;         m2 = m2*2
;         x2 = x2*4
;         b2 = b2*8
;
_ClalTask3:
    MMOV32 MR0, @_m2 ; MR0 = 32 (0x00000020)
    MMOV32 MR1, @_x2 ; MR1 = 64 (0x00000040)
    MMOV32 MR2, @_b2 ; MR2 = -128 (0xFFFFFFFF80)
    MLSL32 MR0, #1 ; MR0 = 64 (0x00000040)
    MLSL32 MR1, #2 ; MR1 = 256 (0x00000100)
    MLSL32 MR2, #3 ; MR2 = -1024 (0xFFFFFC00)
    MMOV32 @_m2, MR0 ; Store results
    MMOV32 @_x2, MR1
    MMOV32 @_b2, MR2
    MSTOP ; end of task
```

### See also

[MADD32 MRa, MRb, MRc](#)  
[MASR32 MRa, #SHIFT](#)  
[MAND32 MRa, MRb, MRc](#)  
[MLSR32 MRa, #SHIFT](#)  
[MOR32 MRa, MRb, MRc](#)  
[MXOR32 MRa, MRb, MRc](#)  
[MSUB32 MRa, MRb, MRc](#)

**MLSR32 MRa, #SHIFT Logical Shift Right**
**Operands**

MRa	CLA floating-point source/destination register (MR0 to MR3)
#SHIFT	Number of bits to shift (1 to 32)

**Opcode**

```
LSW: 0000 0000 0shi ftaa
MSW: 0111 1011 1000 0000
```

**Description**

Logical shift right of MRa by the number of bits indicated. The number of bits can be 1 to 32. Unlike the arithmetic shift (MASR32), the logical shift does not preserve the number's sign bit. Every bit in the operand is moved the specified number of bit positions, and the vacant bit-positions are filled in with zeros

```
MARa(31:0) = Logical Shift Right(MARa(31:0) by #SHIFT bits);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Illustrate the difference between MASR32 and MLSR32
```

```
MMOVIZ MR0, #0xAAAA ; MR0 = 0xAAAA5555
MMOVXI MR0, #0x5555
```

```
MMOV32 MR1, MR0 ; MR1 = 0xAAAA5555
MMOV32 MR2, MR0 ; MR2 = 0xAAAA5555
```

```
MASR32 MR1, #1 ; MR1 = 0xD5552AAA
MLSR32 MR2, #1 ; MR2 = 0x55552AAA
```

```
MASR32 MR1, #1 ; MR1 = 0xEAAA9555
MLSR32 MR2, #1 ; MR2 = 0x2AAA9555
```

```
MASR32 MR1, #6 ; MR1 = 0xFFAAA555
MLSR32 MR2, #6 ; MR2 = 0x00AAA555
```

**See also**

[MADD32 MRa, MRb, MRc](#)  
[MASR32 MRa, #SHIFT](#)  
[MAND32 MRa, MRb, MRc](#)  
[MLSL32 MRa, #SHIFT](#)  
[MOR32 MRa, MRb, MRc](#)  
[MXOR32 MRa, MRb, MRc](#)  
[MSUB32 MRa, MRb, MRc](#)



## MMACF32 MR3, MR2, MRd, MRe, MRf ||MMOV32 MRa, mem32 32-Bit Floating-Point Multiply and Accumulate with Parallel Move

### Operands

MR3	floating-point destination/source register MR3 for the add operation
MR2	CLA floating-point source register MR2 for the add operation
MRd	CLA floating-point destination register (MR0 to MR3) for the multiply operation MRd cannot be the same register as MRa
MRe	CLA floating-point source register (MR0 to MR3) for the multiply operation
MRf	CLA floating-point source register (MR0 to MR3) for the multiply operation
MRa	CLA floating-point destination register for the MMOV32 operation (MR0 to MR3). MRa cannot be MR3 or the same register as MRd.
mem32	32-bit source for the MMOV32 operation

### Opcode

LSW: `mmmm mmmm mmmm mmmm`  
MSW: `0011 ffee ddaa addr`

### Description

Multiply and accumulate the contents of floating-point registers and move from register to memory. The destination register for the MMOV32 cannot be the same as the destination registers for the MMACF32.

```
MR3 = MR3 + MR2;
MRd = MRe * MRf;
MRa = [mem32];
```

### Restrictions

The destination registers for the MMACF32 and the MMOV32 must be unique. That is, MRa cannot be MR3 and MRa cannot be the same register as MRd.

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MMACF32 (add or multiply) generates an underflow condition.
- LVF = 1 if MMACF32 (add or multiply) generates an overflow condition.

MMOV32 sets the NF and ZF flags as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; }
```

### Pipeline

MMACF32 and MMOV32 complete in a single cycle.

**Example 1**

```

; Perform 5 multiply and accumulate operations:
;
; X and Y are 32-bit floating point arrays
;
; 1st multiply: A = X0 * Y0
; 2nd multiply: B = X1 * Y1
; 3rd multiply: C = X2 * Y2
; 4th multiply: D = X3 * Y3
; 5th multiply: E = X3 * Y3
;
; Result = A + B + C + D + E
;
_ClalTask1:
    MMOV16 MAR0, #_X                ; MAR0 points to X array
    MMOV16 MAR1, #_Y                ; MAR1 points to Y array
    MNOP                            ; Delay for MAR0, MAR1 load
    MNOP                            ; Delay for MAR0, MAR1 load
    ; <-- MAR0 valid
    MMOV32 MR0, *MAR0[2]++         ; MR0 = X0, MAR0 += 2
    ; <-- MAR1 valid
    MMOV32 MR1, *MAR1[2]++         ; MR1 = Y0, MAR1 += 2
    ;
    MMPYF32 MR2, MR0, MR1          ; MR2 = A = X0 * Y0
    | MMOV32 MR0, *MAR0[2]++        ; In parallel MR0 = X1, MAR0 += 2
    MMOV32 MR1, *MAR1[2]++        ; MR1 = Y1, MAR1 += 2
    ;
    MMPYF32 MR3, MR0, MR1          ; MR3 = B = X1 * Y1
    | MMOV32 MR0, *MAR0[2]++        ; In parallel MR0 = X2, MAR0 += 2
    MMOV32 MR1, *MAR1[2]++        ; MR1 = Y2, MAR2 += 2
    ;
    MMACF32 MR3, MR2, MR2, MR0, MR1 ; MR3 = A + B, MR2 = C = X2 * Y2
    | MMOV32 MR0, *MAR0[2]++        ; In parallel MR0 = X3
    MMOV32 MR1, *MAR1[2]++        ; MR1 = Y3 M
    ;
    MACF32 MR3, MR2, MR2, MR0, MR1 ; MR3 = (A + B) + C, MR2 = D = X3 * Y3
    | MMOV32 MR0, *MAR0             ; In parallel MR0 = X4
    MMOV32 MR1, *MAR1             ; MR1 = Y4
    ;
    MMPYF32 MR2, MR0, MR1          ; MR2 = E = X4 * Y4
    | MADD32 MR3, MR3, MR2         ; in parallel MR3 = (A + B + C) + D
    ;
    MADD32 MR3, MR3, MR2          ; MR3 = (A + B + C + D) + E
    MMOV32 @_Result, MR3          ; Store the result
    MSTOP                          ; end of task

```

**Example 2**

```

; sum = X0*B0 + X1*B1 + X2*B2 + Y1*A1 + Y2*B2
;
;      X2 = X1
;      X1 = X0
;      Y2 = Y1 ; Y1 = sum
;
_ClaTask2:
    MMOV32    MR0, @_B2      ; MR0 = B2
    MMOV32    MR1, @_X2      ; MR1 = X2
    MMPYF32   MR2, MR1, MR0  ; MR2 = X2*B2
    || MMOV32    MR0, @_B1      ; MR0 = B1
    MMOVD32   MR1, @_X1      ; MR1 = X1, X2 = X1
    MMPYF32   MR3, MR1, MR0  ; MR3 = X1*B1
    || MMOV32    MR0, @_B0      ; MR0 = B0
    MMOVD32   MR1, @_X0      ; MR1 = X0, X1 = X0

; MR3 = X1*B1 + X2*B2, MR2 = X0*B0
; MR0 = A2
    MMACF32   MR3, MR2, MR2, MR1, MR0
    || MMOV32   MR0, @_A2 M

    MOV32    MR1, @_Y2      ; MR1 = Y2

; MR3 = X0*B0 + X1*B1 + X2*B2, MR2 = Y2*A2
; MR0 = A1
    MMACF32   MR3, MR2, MR2, MR1, MR0
    || MMOV32   MR0, @_A1

    MMOVD32   MR1, @_Y1      ; MR1 = Y1, Y2 = Y1
    MADDF32   MR3, MR3, MR2  ; MR3 = Y2*A2 + X0*B0 + X1*B1 + X2*B2
    || MMPYF32   MR2, MR1, MR0 ; MR2 = Y1*A1
    MADDF32   MR3, MR3, MR2  ; MR3 = Y1*A1 + Y2*A2 + X0*B0 + X1*B1 + X2*B2
    MMOV32   @_Y1, MR3      ; Y1 = MR3
    MSTOP
    ; end of task

```

**See also**

[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf](#)

**MMAXF32 MRa, MRb 32-Bit Floating-Point Maximum**
**Operands**

MRa	CLA floating-point source/destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0010 0000
```

**Description**

```
if (MRa < MRb) MRa = MRb;
```

Special cases for the output from the MMAXF32 operation:

- NaN output will be converted to infinity
- A denormalized output will be converted to positive zero.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if (MRa == MRb) {ZF=1; NF=0;}
if (MRa > MRb) {ZF=0; NF=0;}
if (MRa < MRb) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
MMOVIZ MR0, #5.0 ; MR0 = 5.0 (0x40A00000)
MMOVIZ MR1, #-2.0 ; MR1 = -2.0 (0xC0000000)
MMOVIZ MR2, #-1.5 ; MR2 = -1.5 (0xBFC00000)
MMAXF32 MR2, MR1 ; MR2 = -1.5, ZF = NF = 0
MMAXF32 MR1, MR2 ; MR1 = -1.5, ZF = 0, NF = 1
MMAXF32 MR2, MR0 ; MR2 = 5.0, ZF = 0, NF = 1
MAXF32 MR0, MR2 ; MR2 = 5.0, ZF = 1, NF = 0
```

**Example 2**

```
; X is an array of 32-bit floating-point values
; Find the maximum value in an array X
; and store it in Result
;
_ClalTask1:
  MMOV16 MAR1, #_X ; Start address
  MUI16TOF32 MR0, @_len ; Length of the array
  MNOP ; delay for MAR1 load
  MNOP ; delay for MAR1 load
  MMOV32 MR1, *MAR1[2]++ ; MR1 = X0
LOOP
  MMOV32 MR2, *MAR1[2]++ ; MR2 = next element
  MMAXF32 MR1, MR2 ; MR1 = MAX(MR1, MR2)
  MADDF32 MR0, MR0, #-1.0 ; Decrement the counter
  MCMPF32 MR0 #0.0 ; Set/clear flags for MBCNDD
  MNOP
  MNOP
  MNOP
  MBCNDD LOOP, NEQ ; Branch if not equal to zero
  MMOV32 @_Result, MR1 ; Always executed
  MNOP ; Always executed
  MNOP ; Always executed
  MSTOP ; End of task
```

**See also**

[MCMPF32 MRa, MRb](#)  
[MCMPF32 MRa, #16FHi](#)  
[MMAXF32 MRa, #16FHi](#)  
[MMINF32 MRa, MRb](#)  
[MMINF32 MRa, #16FHi](#)

**MMAXF32 MRa, #16FHi 32-Bit Floating-Point Maximum**
**Operands**

MRa	CLA floating-point source/destination register (MR0 to MR3)
#16FHi	A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0.

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 1001 0000 00aa
```

**Description**

Compare MRa with the floating-point value represented by the immediate operand. If the immediate value is larger, then load it into MRa.

```
if(MRa < #16FHi:0) MRa = #16FHi:0;
```

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. This addressing mode is most useful for constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

Special cases for the output from the MMAXF32 operation:

- NaN output will be converted to infinity
- A denormalized output will be converted to positive zero.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == #16FHi:0) {ZF=1; NF=0;}
if(MRa > #16FHi:0) {ZF=0; NF=0;}
if(MRa < #16FHi:0) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ MR0, #5.0 ; MR0 = 5.0 (0x40A00000)
MMOVIZ MR1, #4.0 ; MR1 = 4.0 (0x40800000)
MMOVIZ MR2, #-1.5 ; MR2 = -1.5 (0xBFC00000)
MMAXF32 MR0, #5.5 ; MR0 = 5.5, ZF = 0, NF = 1
MMAXF32 MR1, #2.5 ; MR1 = 4.0, ZF = 0, NF = 0
MMAXF32 MR2, #-1.0 ; MR2 = -1.0, ZF = 0, NF = 1
MMAXF32 MR2, #-1.0 ; MR2 = -1.5, ZF = 1, NF = 0
```

**See also**

[MMAXF32 MRa, MRb](#)  
[MMINF32 MRa, MRb](#)  
[MMINF32 MRa, #16FHi](#)

## MMINF32 MRa, MRb 32-Bit Floating-Point Minimum

### Operands

MRa	CLA floating-point source/destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

### Opcode

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0100 0000
```

### Description

```
if (MRa > MRb) MRa = MRb;
```

Special cases for the output from the MMINF32 operation:

- NaN output will be converted to infinity
- A denormalized output will be converted to positive zero.

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if (MRa == MRb) {ZF=1; NF=0;}
if (MRa > MRb) {ZF=0; NF=0;}
if (MRa < MRb) {ZF=0; NF=1;}
```

### Pipeline

This is a single-cycle instruction.

### Example 1

```
MMOVIZ MR0, #5.0 ; MR0 = 5.0 (0x40A00000)
MMOVIZ MR1, #4.0 ; MR1 = 4.0 (0x40800000)
MMOVIZ MR2, #-1.5 ; MR2 = -1.5 (0xBFC00000)
MMINF32 MR0, MR1 ; MR0 = 4.0, ZF = 0, NF = 0
MMINF32 MR1, MR2 ; MR1 = -1.5, ZF = 0, NF = 0
MMINF32 MR2, MR1 ; MR2 = -1.5, ZF = 1, NF = 0
MMINF32 MR1, MR0 ; MR2 = -1.5, ZF = 0, NF = 1
```

### Example 2

```
;
; X is an array of 32-bit floating-point values
; Find the minimum value in an array X
; and store it in Result
;

_ClalTask1:
MMOVI16   MAR1, #_X           ; Start address
MUI16TOF32 MR0, @_len        ; Length of the array
MNOP                               ; delay for MAR1 load
MNOP                               ; delay for MAR1 load
MMOV32    MR1, *MAR1[2]++    ; MR1 = X0
LOOP
MMOV32    MR2, *MAR1[2]++    ; MR2 = next element
MMINF32   MR1, MR2           ; MR1 = MAX(MR1, MR2)
MADDF32   MR0, MR0, #-1.0    ; Decrement the counter
MCMPIF32  MR0 #0.0          ; Set/clear flags for MBCNDD
MNOP
MNOP
MNOP
MBCNDD    LOOP, NEQ          ; Branch if not equal to zero
MMOV32    @_Result, MR1     ; Always executed
MNOP                               ; Always executed
MNOP                               ; Always executed
MSTOP                               ; End of task
```

**See also**

[MMAXF32 MRa, MRb](#)  
[MMAXF32 MRa, #16FHi](#)  
[MMINF32 MRa, #16FHi](#)



## MMINF32 MRa, #16FHi 32-Bit Floating-Point Minimum

### Operands

MRa	floating-point source/destination register (MR0 to MR3)
#16FHi	A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0.

### Opcode

```
LSW: IIII IIII IIII IIII
MSW: 0111 1001 0100 00aa
```

### Description

Compare MRa with the floating-point value represented by the immediate operand. If the immediate value is smaller, then load it into MRa.

```
if(MRa > #16FHi:0) MRa = #16FHi:0;
```

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. This addressing mode is most useful for constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

Special cases for the output from the MMINF32 operation:

- NaN output will be converted to infinity
- A denormalized output will be converted to positive zero.

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == #16FHi:0) {ZF=1; NF=0;}
if(MRa > #16FHi:0) {ZF=0; NF=0;}
if(MRa < #16FHi:0) {ZF=0; NF=1;}
```

### Pipeline

This is a single-cycle instruction.

### Example

```
MMOVIZ MR0, #5.0 ; MR0 = 5.0 (0x40A00000)
MMOVIZ MR1, #4.0 ; MR1 = 4.0 (0x40800000)
MMOVIZ MR2, #-1.5 ; MR2 = -1.5 (0xBFC00000)
MMINF32 MR0, #5.5 ; MR0 = 5.0, ZF = 0, NF = 1
MMINF32 MR1, #2.5 ; MR1 = 2.5, ZF = 0, NF = 0
MMINF32 MR2, #-1.0 ; MR2 = -1.5, ZF = 0, NF = 1
MMINF32 MR2, #-1.5 ; MR2 = -1.5, ZF = 1, NF = 0
```

### See also

[MMAXF32 MRa, #16FHi](#)  
[MMAXF32 MRa, MRb](#)  
[MMINF32 MRa, MRb](#)

### MMOV16 MARx, MRa, #16I Load the Auxiliary Register with MRa + 16-bit Immediate Value

**Operands**

MARx	Auxiliary register MAR0 or MAR1
MRa	CLA Floating-point register (MR0 to MR3)
#16I	16-bit immediate value

**Opcode**

LSW: I I I I I I I I I I I I I I (opcode of MMOV16 MAR0, MRa, #16I)  
 MSW: 0111 1111 1101 00AA

LSW: I I I I I I I I I I I I I I (opcode of MMOV16 MAR1, MRa, #16I)  
 MSW: 0111 1111 1111 00AA

**Description**

Load the auxiliary register, MAR0 or MAR1, with MRa(15:0) + 16-bit immediate value. Refer to the pipeline section for important information regarding this instruction.

$$MARx = MRa(15:0) + \#16I;$$

**Flags**

This instruction does not modify flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction. The load of MAR0 or MAR1 will occur in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing will occur in the D2 phase of the pipeline. Therefore the following applies when loading the auxiliary registers:

- I1 and I2**

The two instructions following MMOV16 will use MAR0/MAR1 before the update occurs. Thus these two instructions will use the old value of MAR0 or MAR1.
- I3**

Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus I3 cannot use the auxiliary register or there will be a conflict. In the case of a conflict, the update due to address-mode post increment will win and the auxiliary register will not be updated with #\_X.
- I4**

Starting with the 4th instruction MAR0 or MAR1 will be the new value loaded with MMOV16.

; Assume MAR0 is 50, MR0 is 10, and #\_X is 20

```

MMOV16 MAR0, MR0, #_X      ; Load MAR0 with address of X (20) + MR0 (10)
<Instruction 1>             ; I1 Will use the old value of MAR0 (50)
<Instruction 2>             ; I2 Will use the old value of MAR0 (50)
<Instruction 3>             ; I3 Cannot use MAR0
<Instruction 4>             ; I4 Will use the new value of MAR0 (30)
<Instruction 5>             ; I5
  
```

**Table 5-17. Pipeline Activity For MMOV16 MARx, MRa , #16I**

Instruction	F1	F2	D1	D2	R1	R2	E	W
MMOV16 MAR0, MR0, #_X	MMOV16							
I1	I1	MMOV16						
I2	I2	I1	MMOV16					
I3	I3	I2	I1	MMOV16				
I4	I4	I3	I2	I1	MMOV16			
I5	I5	I4	I3	I2	I1	MMOV16		
I6	I6	I5	I4	I3	I2	I1	MMOV16	

**Example 1**

```

; Calculate an offset into a sin/cos table
;
_ClalTask1:
    MMOV32 MR0,@_rad                ; MR0 = rad
    MMOV32 MR1,@_TABLE_SIZEDivTwoPi ; MR1 = TABLE_SIZE/(2*Pi)
    MMPYF32 MR1,MR0,MR1             ; MR1 = rad* TABLE_SIZE/(2*Pi)
||  MMOV32 MR2,@_TABLE_MASK         ; MR2 = TABLE_MASK
    MF32TOI32 MR3,MR1               ; MR3 = K=int(rad*TABLE_SIZE/(2*Pi))
    MAND32 MR3,MR3,MR2              ; MR3 = K & TABLE_MASK
    MLSL32 MR3,#1                   ; MR3 = K * 2

    MMOV16 MAR0,MR3,#_Cos0          ; MAR0 K*2+addr of table.Cos0
    MFRACF32 MR1,MR1                ; I1
    MMOV32 MR0,@_TwoPiDivTABLE_SIZE ; I2
    MMPYF32 MR1,MR1,MR0             ; I3
||  MMOV32 MR0,@_Coef3

    MMOV32 MR2,*MAR0[#-64]++        ; MR2 = *MAR0, MAR0 += (-64)
    ...
    ...
    MSTOP ; end of task

```

**Example 2**

```

; This task logs the last NUM_DATA_POINTS
; ADCRESULT1 values in the array VoltageCLA
;
; When the last element in the array has been
; filled, the task will go back to the
; the first element.
;
; Before starting the ADC conversions, force
; Task 8 to initialize the ConversionCount to zero
;
; The ADC is set to sample (acquire) for 15 SYSCLK cycles
; or 75ns. After the capacitor has captured the analog
; value, the ADC will trigger this task early.
; It takes 10.5 ADCCLKs to complete a conversion,
; the ADCCLK being SYSCLK/4
;   T_sys = 1/200MHz = 5ns
;   T_adc = 4*T_sys = 20ns
; The ADC will take 10.5 * 4 or 42 SYSCLK cycles to complete
; a conversion. The ADC result register may be read on the
; 36th instruction after the task begins.
;
_ClalTask2:
    .asg          0, N
    .loop
    MNOP                                     ;I1 - I28 Wait till I36 to read result
    .eval        N + 1, N
    .break       N = 28
    .endloop
    MMOVZ16 MR0, @_ConversionCount          ;I29 Current Conversion
    MMOV16 MAR1, MR0, #_VoltageCLA         ;I30 Next array location
    MUI16TOF32 MR0, MR0                    ;I31 Convert count to float32
    MADDF32 MR0, MR0, #1.0                 ;I32 Add 1 to conversion count
    MCMFP32 MR0, #NUM_DATA_POINTS.0       ;I33 Compare count to max
    MF32TOUI16 MR0, MR0                    ;I34 Convert count to Uint16
    MNOP                                     ;I35 Wait till I36 to read result
    MMOVZ16 MR2, @_AdcaResultRegs.ADCRESULT1 ;I36 Read ADCRESULT1
    MMOV16 *MAR1, MR2                       ; Store ADCRESULT1
    MBCNDD _RestartCount, GEQ              ; If count >= NUM_DATA_POINTS
    MMOVIZ MR1, #0.0                       ; Always executed: MR1=0
    MNOP
    MNOP
    MMOV16 @_ConversionCount, MR0          ; If branch not taken
    MSTOP                                  ; store current count

```

```
_RestartCount
    MMOV16    @_ConversionCount, MR1        ; If branch taken, restart count
    MSTOP                                         ; end of task

; This task initializes the ConversionCount
; to zero
;
_Cla1Task8:
    MMOVIZ    MR0, #0.0
    MMOV16    @_ConversionCount, MR0
    MSTOP
_ClaT8End:
```

**See also**

**MMOV16 MARx, mem16 Load MAR1 with 16-bit Value**
**Operands**

MARx	CLA auxiliary register MAR0 or MAR1
mem16	16-bit destination memory accessed using one of the available addressing modes

**Opcode**

LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 MAR0, mem16)  
MSW: 0111 0110 0000 addr

LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 MAR1, mem16)  
MSW: 0111 0110 0100 addr

**Description**

Load MAR0 or MAR1 with the 16-bit value pointed to by mem16. Refer to the pipeline section for important information regarding this instruction.

MAR1 = [mem16];

**Flags**

No flags MSTF flags are affected.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction. The load of MAR0 or MAR1 will occur in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing will occur in the D2 phase of the pipeline. Therefore the following applies when loading the auxiliary registers:

- **I1 and I2**

The two instructions following MMOV16 will use MAR0/MAR1 before the update occurs. Thus these two instructions will use the old value of MAR0 or MAR1.

- **I3**

Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus I3 cannot use the auxiliary register or there will be a conflict. In the case of a conflict, the update due to address-mode post increment will win send the auxiliary register will not be updated with #\_X.

- **I4**

Starting with the 4th instruction MAR0 or MAR1 will be the new value loaded with MMOV16.

; Assume MAR0 is 50 and @\_X is 20

```
MMOV16 MAR0, @_X ; Load MAR0 with the contents of X (20)
<Instruction 1> ; I1 Will use the old value of MAR0 (50)
<Instruction 2> ; I2 Will use the old value of MAR0 (50)
<Instruction 3> ; I3 Cannot use MAR0
<Instruction 4> ; I4 Will use the new value of MAR0 (20)
<Instruction 5> ; I5
....
```

**Table 5-18. Pipeline Activity For MMOV16 MAR0/MAR1, mem16**

Instruction	F1	F2	D1	D2	R1	R2	E	W
MMOV16 MAR0, @_X	MMOV16							
I1	I1	MMOV16						
I2	I2	I1	MMOV16					
I3	I3	I2	I1	MMOV16				
I4	I4	I3	I2	I1	MMOV16			
I5	I5	I4	I3	I2	I1	MMOV16		
I6	I6	I5	I4	I3	I2	I1	MMOV16	

**Example**

```

; This task logs the last NUM_DATA_POINTS
; ADCRESULT1 values in the array VoltageCLA
;
; When the last element in the array has been
; filled, the task will go back to the
; the first element.
;
; Before starting the ADC conversions, force
; Task 8 to initialize the ConversionCount to zero
;
; The ADC is set to sample (acquire) for 15 SYSCLK cycles
; or 75ns. After the capacitor has captured the analog
; value, the ADC will trigger this task early.
; It takes 10.5 ADCCLKs to complete a conversion,
; the ADCCLK being SYSCLK/4
;   T_sys = 1/200MHz = 5ns
;   T_adc = 4*T_sys = 20ns
; The ADC will take 10.5 * 4 or 42 SYSCLK cycles to complete
; a conversion. The ADC result register may be read on the
; 36th instruction after the task begins.
;
_Cla1Task2:
    .asg      0, N
    .loop
    MNOPI
;I1 - I28 Wait till I36 to read result
    .eval    N + 1, N
    .break   N = 28
    .endloop
    MMOVZ16  MR0, @_ConversionCount           ;I29 Current Conversion
    MMOV16   MAR1, MR0, #_VoltageCLA         ;I30 Next array location
    MUI16TOF32 MR0, MR0                     ;I31 Convert count to float32
    MADD32   MR0, MR0, #1.0                 ;I32 Add 1 to conversion count
    MCMPF32  MR0, #NUM_DATA_POINTS.0       ;I33 Compare count to max
    MF32TOUI16 MR0, MR0                    ;I34 Convert count to Uint16
    MNOPI
    MMOVZ16  MR2, @_AdcaResultRegs.ADCRESULT1 ;I36 Read ADCRESULT1
    MMOV16   *MAR1, MR2                     ; Store ADCRESULT1
    MBCNDD   _RestartCount, GEQ             ; If count >= NUM_DATA_POINTS
    MMOVIZ   MR1, #0.0                      ; Always executed: MR1=0
    MNOPI
    MNOPI
    MMOV16   @_ConversionCount, MR0         ; If branch not taken
    MSTOP    ; store current count

    _RestartCount
    MMOV16   @_ConversionCount, MR1         ; If branch taken, restart count
    MSTOP    ; end of task

; This task initializes the ConversionCount
; to zero
;
_Cla1Task8:
    MMOVIZ   MR0, #0.0
    MMOV16   @_ConversionCount, MR0
    MSTOP
_ClaT8End:

```

**See also**

**MMOV16 mem16, MARx *Move 16-Bit Auxiliary Register Contents to Memory***
**Operands**

mem16	16-bit destination memory accessed using one of the available addressing modes
MARx	CLA auxiliary register MAR0 or MAR1

**Opcode**

LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 mem16, MAR0)  
 MSW: 0111 0110 1000 addr

LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 mem16, MAR1)  
 MSW: 0111 0110 1100 addr

**Description**

Store the contents of MAR0 or MAR1 in the 16-bit memory location pointed to by mem16.

[mem16] = MAR0;

**Flags**

No flags MSTF flags are affected.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**
**See also**

**MMOV16 mem16, MRa *Move 16-Bit Floating-Point Register Contents to Memory***
**Operands**

mem16	16-bit destination memory accessed using one of the available addressing modes
MRa	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 11aa addr
```

**Description**

Move 16-bit value from the lower 16-bits of the floating-point register (MRa(15:0)) to the location pointed to by mem16.

```
[mem16] = MRa(15:0);
```

**Flags**

No flags MSTF flags are affected.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; This task logs the last NUM_DATA_POINTS
; ADCRESULT1 values in the array VoltageCLA
;
; When the last element in the array has been
; filled, the task will go back to the
; the first element.
;
; Before starting the ADC conversions, force
; Task 8 to initialize the ConversionCount to zero
;
; The ADC is set to sample (acquire) for 15 SYSCLK cycles
; or 75ns. After the capacitor has captured the analog
; value, the ADC will trigger this task early.
; It takes 10.5 ADCCLKs to complete a conversion,
; the ADCCLK being SYSCLK/4
; T_sys = 1/200MHz = 5ns
; T_adc = 4*T_sys = 20ns
; The ADC will take 10.5 * 4 or 42 SYSCLK cycles to complete
; a conversion. The ADC result register may be read on the
; 36th instruction after the task begins.
;
_ClalTask2:
    .asg      0, N
    .loop
    MNOPI
;I1 - I28 Wait till I36 to read result
    .eval    N + 1, N
    .break   N = 28
    .endloop
MMOVZ16    MR0, @_ConversionCount           ;I29 Current Conversion
MMOV16     MAR1, MR0, #_VoltageCLA          ;I30 Next array location
MUI16TOF32 MR0, MR0                         ;I31 Convert count to float32
MADDF32    MR0, MR0, #1.0                   ;I32 Add 1 to conversion count
MCMFP32    MR0, #NUM_DATA_POINTS.0         ;I33 Compare count to max
MF32TOUI16 MR0, MR0                         ;I34 Convert count to Uint16
MNOPI
MMOVZ16    MR2, @_AdcaResultRegs.ADCRESULT1 ;I36 Read ADCRESULT1
MMOV16     *MAR1, MR2                       ; Store ADCRESULT1
MBCNDD     _RestartCount, GEQ              ; If count >= NUM_DATA_POINTS
MMOVIZ     MR1, #0.0                       ; Always executed: MR1=0
MNOPI
MNOPI
MMOV16     @_ConversionCount, MR0          ; If branch not taken
MSTOP     ; store current count
```



```
_RestartCount
    MMOV16    @_ConversionCount, MR1        ; If branch taken, restart count
    MSTOP                                         ; end of task

; This task initializes the ConversionCount
; to zero
;
_Cla1Task8:
    MMOVIZ    MR0, #0.0
    MMOV16    @_ConversionCount, MR0
    MSTOP
_ClaT8End:
```

**See also**

[MMOVIZ MRa, #16FHi](#)  
[MMOVXI MRa, #16FLoHex](#)

## MMOV32 mem32, MRa *Move 32-Bit Floating-Point Register Contents to Memory*

### Operands

MRa	floating-point register (MR0 to MR3)
mem32	32-bit destination memory accessed using one of the available addressing modes

### Opcode

LSW: mmmmm mmmmm mmmmm mmmmm  
 MSW: 0111 0100 11aa addr

### Description

Move from MRa to 32-bit memory location indicated by mem32.

[mem32] = MRa;

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

No flags affected.

### Pipeline

This is a single-cycle instruction.

### Example

```

; Perform 5 multiply and accumulate operations:
;
; X and Y are 32-bit floating point arrays;
; 1st multiply: A = X0 * Y0
; 2nd multiply: B = X1 * Y1
; 3rd multiply: C = X2 * Y2
; 4th multiply: D = X3 * Y3
; 5th multiply: E = X3 * Y3;
; Result = A + B + C + D + E
;
_ClalTask1:
  MMOV16    MAR0, #_X           ; MAR0 points to X array
  MMOV16    MAR1, #_Y           ; MAR1 points to Y array
  MNOP      ; Delay for MAR0, MAR1 load
  MNOP      ; Delay for MAR0, MAR1 load
  ; <-- MAR0 valid
  MMOV32    MR0, *MAR0[2]++     ; MR0 = X0, MAR0 += 2
  ; <-- MAR1 valid
  MMOV32    MR1, *MAR1[2]++     ; MR1 = Y0, MAR1 += 2
  MMPYF32   MR2, MR0, MR1       ; MR2 = A = X0 * Y0
  | MMOV32   MR0, *MAR0[2]++     ; In parallel MR0 = X1, MAR0 += 2
  MMOV32    MR1, *MAR1[2]++     ; MR1 = Y1, MAR1 += 2
  MMPYF32   MR3, MR0, MR1       ; MR3 = B = X1 * Y1
  | MMOV32   MR0, *MAR0[2]++     ; In parallel MR0 = X2, MAR0 += 2
  MMOV32    MR1, *MAR1[2]++     ; MR1 = Y2, MAR2 += 2

  MMACF32   MR3, MR2, MR2, MR0, MR1 ; MR3 = A + B, MR2 = C = X2 * Y2
  | MMOV32   MR0, *MAR0[2]++     ; In parallel MR0 = X3
  MMOV32    MR1, *MAR1[2]++     ; MR1 = Y3

  MMACF32   MR3, MR2, MR2, MR0, MR1 ; MR3 = (A + B) + C, MR2 = D = X3 * Y3
  | MMOV32   MR0, *MAR0
  MMOV32    MR1, *MAR1           ; MR1 = Y4
  MMPYF32   MR2, MR0, MR1       ; MR2 = E = X4 * Y4
  | MADDF32  MR3, MR3, MR2       ; in parallel MR3 = (A + B + C) + D
  MADDF32   MR3, MR3, MR2       ; MR3 = (A + B + C + D) + E
  MMOV32    @_Result, MR3       ; Store the result MSTOP ; end of task
  
```

### See also

[MMOV32 mem32, MSTF](#)

**MMOV32 mem32, MSTF** *Move 32-Bit MSTF Register to Memory*
**Operands**

MSTF	floating-point status register
mem32	32-bit destination memory

**Opcode**

LSW: mmmm mmmm mmmm mmmm  
 MSW: 0111 0111 0100 addr

**Description**

Copy the CLA's floating-point status register, MSTF, to memory.

[mem32] = MSTF;

**Flags**

This instruction does not modify flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**
**See also**

[MMOV32 mem32, MRa](#)

**MMOV32 MRa, mem32 {, CNDF} *Conditional 32-Bit Move***
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
mem32	32-bit memory location accessed using one of the available addressing modes
CNDF	optional condition.

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 00cn dfaa addr
```

**Description**

If the condition is true, then move the 32-bit value referenced by mem32 to the floating-point register indicated by MRa.

```
if (CNDF == TRUE) MRa = [mem32];
```

CNDF is one of the following conditions:

Encode <sup>(1)</sup>	CNDF	Description	MSTF Flags Tested
0000	NEQ	Not equal to zero	ZF == 0
0001	EQ	Equal to zero	ZF == 1
0010	GT	Greater than zero	ZF == 0 AND NF == 0
0011	GEQ	Greater than or equal to zero	NF == 0
0100	LT	Less than zero	NF == 1
0101	LEQ	Less than or equal to zero	ZF == 1 OR NF == 1
1010	TF	Test flag set	TF == 1
1011	NTF	Test flag not set	TF == 0
1100	LU	Latched underflow	LUF == 1
1101	LV	Latched overflow	LVF == 1
1110	UNC	Unconditional	None
1111	UNCF <sup>(2)</sup>	Unconditional with flag modification	None

<sup>(1)</sup> Values not shown are reserved.

<sup>(2)</sup> This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

```
if (CNDF == UNCF)
{
    NF = MRa(31);
    ZF = 0;
    if (MRa(30:23) == 0) { ZF = 1; NF = 0; }
}
else No flags modified;
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```

; Given A, B, X, M1 and M2 are 32-bit floating-point
; numbers
;
; if(A == B) calculate Y = X*M1
; if(A! = B) calculate Y = X*M2
;
_Cla1Task5:
    MMOV32    MR0, @_A
    MMOV32    MR1, @_B
    MCMFP32   MR0, MR1
    MMOV32    MR2, @_M1, EQ ; if A == B, MR2 = M1
                          ; Y = M1*X
    MMOV32    MR2, @_M2, NEQ ; if A! = B, MR2 = M2
                          ; Y = M2*X

    MMOV32    MR3, @_X
    MMPYF32   MR3, MR2, MR3 ; Calculate Y
    MMOV32    @_Y, MR3      ; Store Y
    MSTOP
                          ; end of task

```

**See also**

[MMOV32 MRa, MRb {, CNDF}](#)  
[MMOVD32 MRa, mem32](#)

**MMOV32 MRa, MRb {, CNDF} Conditional 32-Bit Move**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)
CNDF	optional condition.

**Opcode**

```
LSW: 0000 0000 cndf bbaa
MSW: 0111 1010 1100 0000
```

**Description**

If the condition is true, then move the 32-bit value in MRb to the floating-point register indicated by MRa.

```
if (CNDF == TRUE) MRa = MRb;
```

CNDF is one of the following conditions:

Encode <sup>(3)</sup>	CNDF	Description	MSTF Flags Tested
0000	NEQ	Not equal to zero	ZF == 0
0001	EQ	Equal to zero	ZF == 1
0010	GT	Greater than zero	ZF == 0 AND NF == 0
0011	GEQ	Greater than or equal to zero	NF == 0
0100	LT	Less than zero	NF == 1
0101	LEQ	Less than or equal to zero	ZF == 1 OR NF == 1
1010	TF	Test flag set	TF == 1
1011	NTF	Test flag not set	TF == 0
1100	LU	Latched underflow	LUF == 1
1101	LV	Latched overflow	LVF == 1
1110	UNC	Unconditional	None
1111	UNCF <sup>(4)</sup>	Unconditional with flag modification	None

<sup>(3)</sup> Values not shown are reserved.

<sup>(4)</sup> This is the default operation if no CNDF field is specified. This condition will allow the ZF, and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

```
if (CNDF == UNCF)
{
    NF = MRa(31); ZF = 0;
    if (MRa(30:23) == 0) {ZF = 1; NF = 0;}
}
else No flags modified;
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```

; Given: X = 8.0
;         Y = 7.0
;         A = 2.0
;         B = 5.0
; _ClaTask1
  MMOV32 MR3, @_X      ; MR3 = X = 8.0
  MMOV32 MR0, @_Y      ; MR0 = Y = 7.0
  MMAXF32 MR3, MR0     ; ZF = 0, NF = 0, MR3 = 8.0
  MMOV32 MR1, @_A, GT  ; true, MR1 = A = 2.0
  MMOV32 MR1, @_B, LT  ; false, does not load MR1
  MMOV32 MR2, MR1, GT  ; true, MR2 = MR1 = 2.0
  MMOV32 MR2, MR0, LT  ; false, does not load MR2
  MSTOP

```

**See also**

[MMOV32 MRa, mem32 {,CNDF}](#)

**MMOV32 MSTF, mem32** *Move 32-Bit Value from Memory to the MSTF Register*
**Operands**

MSTF	CLA status register
mem32	32-bit source memory location

**Opcode**

LSW: mmmm mmmm mmmm mmmm  
 MSW: 0111 0111 0000 addr

**Description**

Move from memory to the CLA's status register MSTF. This instruction is most useful when nesting function calls (via MCCNDD).

MSTF = [mem32];

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	Yes	Yes	Yes	Yes	Yes

Loading the status register will overwrite all flags and the RPC field. The MEALLOW field is not affected.

**Pipeline**

This is a single-cycle instruction.

**Example**
**See also**

[MMOV32 mem32, MSTF](#)



## MMOVD32 MRa, mem32 *Move 32-Bit Value from Memory with Data Copy*

### Operands

MRa	CLA floating-point register (MR0 to MR3)
mem32	32-bit memory location accessed using one of the available addressing modes

### Opcode

LSW: mmmm mmmm mmmm mmmm  
MSW: 0111 0100 00aa addr

### Description

Move the 32-bit value referenced by mem32 to the floating-point register indicated by MRa.

```
MRa = [mem32];
[mem32+2] = [mem32];
```

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0){ ZF = 1; NF = 0; }
```

### Pipeline

This is a single-cycle instruction.

### Example

```
; sum = X0*B0 + X1*B1 + X2*B2 + Y1*A1 + Y2*B2
;
;      X2 = X1
;      X1 = X0
;      Y2 = Y1
;      Y1 = sum
;
_ClalTask2:
    MMOV32 MR0, @_B2      ; MR0 = B2
    MMOV32 MR1, @_X2      ; MR1 = X2
    MMPYF32 MR2, MR1, MR0 ; MR2 = X2*B2
||  MMOV32 MR0, @_B1      ; MR0 = B1
    MMOVD32 MR1, @_X1     ; MR1 = X1, X2 = X1
    MMPYF32 MR3, MR1, MR0 ; MR3 = X1*B1
||  MMOV32 MR0, @_B0      ; MR0 = B0
    MMOVD32 MR1, @_X0     ; MR1 = X0, X1 = X0

; MR3 = X1*B1 + X2*B2, MR2 = X0*B0
; MR0 = A2
    MMACF32 MR3, MR2, MR2, MR1, MR0
||  MMOV32 MR0, @_A2

    MMOV32 MR1, @_Y2      ; MR1 = Y2

; MR3 = X0*B0 + X1*B1 + X2*B2, MR2 = Y2*A2
; MR0 = A1
    MMACF32 MR3, MR2, MR2, MR1, MR0
||  MMOV32 MR0, @_A1

    MMOVD32 MR1, @_Y1     ; MR1 = Y1, Y2 = Y1
    MADD32 MR3, MR3, MR2  ; MR3 = Y2*A2 + X0*B0 + X1*B1 + X2*B2
||  MMPYF32 MR2, MR1, MR0 ; MR2 = Y1*A1
    MADD32 MR3, MR3, MR2  ; MR3 = Y1*A1 + Y2*A2 + X0*B0 + X1*B1 + X2*B2
    MMOV32 @_Y1, MR3      ; Y1 = MR3
    MSTOP                  ; end of task
```

### See also

[MMOV32 MRa, mem32 {,CNDF}](#)

**MMOVF32 MRa, #32F Load the 32-Bits of a 32-Bit Floating-Point Register**
**Operands**

This instruction is an alias for MMOVIZ and MMOVXI instructions. The second operand is translated by the assembler such that the instruction becomes:

```
MMOVIZ MRa, #16FHiHex MMOVXI MRa, #16FLoHex
```

```
MRa          CLA floating-point destination register (MR0 to MR3)
```

```
#32F        immediate float value represented in floating-point representation
```

**Opcode**

```
LSW: I I I I I I I I I I I I I I (opcode of MMOVIZ MRa, #16FHiHex)
```

```
MSW: 0111 1000 0100 00aa
```

```
LSW: I I I I I I I I I I I I I I (opcode of MMOVXI MRa, #16FLoHex)
```

```
MSW: 0111 1000 1000 00aa
```

**Description**

Note: This instruction accepts the immediate operand only in floating-point representation. To specify the immediate value as a hex value (IEEE 32-bit floating-point format) use the MOVI32 MRa, #32FHex instruction.

Load the 32-bits of MRa with the immediate float value represented by #32F.

#32F is a float value represented in floating-point representation. The assembler will only accept a float value represented in floating-point representation. That is, 3.0 can only be represented as #3.0. #0x40400000 will result in an error.

```
MRa = #32F;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

Depending on #32FH, this instruction takes one or two cycles. If all of the lower 16-bits of the IEEE 32-bit floating-point format of #32F are zeros, then the assembler will convert MMOV32 into only MMOVIZ instruction. If the lower 16-bits of the IEEE 32-bit floating-point format of #32F are not zeros, then the assembler will convert MMOV32 into MMOVIZ and MMOVXI instructions.

**Example**

```
MMOVF32 MR1, #3.0      ; MR1 = 3.0 (0x40400000)
                       ; Assembler converts this instruction as
                       ; MMOVIZ MR1, #0x4040

MMOVF32 MR2, #0.0      ; MR2 = 0.0 (0x00000000)
                       ; Assembler converts this instruction as
                       ; MMOVIZ MR2, #0x0

MMOVF32 MR3, #12.265   ; MR3 = 12.625 (0x41443D71)
                       ; Assembler converts this instruction as
                       ; MMOVIZ MR3, #0x4144
                       ; MMOVXI MR3, #0x3D71
```

**See also**

[MMOVIZ MRa, #16FHi](#)  
[MMOVXI MRa, #16FLoHex](#)  
[MMOVI32 MRa, #32FHex](#)

## MMOVI16 MARx, #16I Load the Auxiliary Register with the 16-Bit Immediate Value

### Operands

MARx	Auxiliary register MAR0 or MAR1
#16I	16-bit immediate value

### Opcode

LSW: I I I I I I I I I I I I I I (opcode of MMOVI16 MAR0, #16I)  
MSW: 0111 1111 1100 0000

LSW: I I I I I I I I I I I I I I (opcode of MMOVI16 MAR1, #16I)  
MSW: 0111 1111 1110 0000

### Description

Load the auxiliary register, MAR0 or MAR1, with a 16-bit immediate value. Refer to the pipeline section for important information regarding this instruction.

MARx = #16I;

### Flags

This instruction does not modify flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

### Pipeline

This is a single-cycle instruction. The immediate load of MAR0 or MAR1 will occur in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing will occur in the D2 phase of the pipeline. Therefore the following applies when loading the auxiliary registers:

- **I1 and I2**

The two instructions following MMOVI16 will use MAR0/MAR1 before the update occurs. Thus these two instructions will use the old value of MAR0 or MAR1.

- **I3**

Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus I3 cannot use the auxiliary register or there will be a conflict. In the case of a conflict, the update due to address-mode post increment will win and the auxiliary register will not be updated with #\_X.

- **I4**

Starting with the 4th instruction MAR0 or MAR1 will be the new value loaded with MMOVI16.

; Assume MAR0 is 50 and #\_X is 20

```
MMOVI16 MAR0, #_X           ; Load MAR0 with address of X (20)
<Instruction 1>             ; I1 Will use the old value of MAR0 (50)
<Instruction 2>             ; I2 Will use the old value of MAR0 (50)
<Instruction 3>             ; I3 Cannot use MAR0
<Instruction 4>             ; I4 Will use the new value of MAR0 (20)
<Instruction 5>             ; I5
.....
```

**Table 5-19. Pipeline Activity For MMOVI16 MAR0/MAR1, #16I**

Instruction	F1	F2	D1	D2	R1	R2	E	W
MMOVI16 MAR0, #_X	MMOVI16							
I1	I1	MMOVI16						
I2	I2	I1	MMOVI16					
I3	I3	I2	I1	MMOVI16				
I4	I4	I3	I2	I1	MMOVI16			
I5	I5	I4	I3	I2	I1	MMOVI16		
I6	I6	I5	I4	I3	I2	I1	MMOVI16	

**MMOVI32 MRa, #32FHex** *Load the 32-Bits of a 32-Bit Floating-Point Register with the Immediate*
**Operands**

MRa	floating-point register (MR0 to MR3)
#32FHex	A 32-bit immediate value that represents an IEEE 32-bit floating-point value.

This instruction is an alias for MMOVIZ and MMOVXI instructions. The second operand is translated by the assembler such that the instruction becomes:

```
MMOVIZ MRa, #16FHiHex
MMOVXI MRa, #16FLoHex
```

**Opcode**

```
LSW: IIII IIII IIII IIII (opcode of MMOVIZ MRa, #16FHiHex)
MSW: 0111 1000 0100 00aa
```

```
LSW: IIII IIII IIII IIII (opcode of MMOVXI MRa, #16FLoHex)
MSW: 0111 1000 1000 00aa
```

**Description**

Note: This instruction only accepts a hex value as the immediate operand. To specify the immediate value with a floating-point representation use the MMOVF32 MRa, #32F instruction.

Load the 32-bits of MRa with the immediate 32-bit hex value represented by #32FHex.

#32FHex is a 32-bit immediate hex value that represents the IEEE 32-bit floating-point value of a floating-point number. The assembler will only accept a hex immediate value. That is, 3.0 can only be represented as #0x40400000. #3.0 will result in an error.

```
MRa = #32FHex;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

Depending on #32FHex, this instruction takes one or two cycles. If all of the lower 16-bits of #32FHex are zeros, then assembler will convert MOVIZ to the MMOVIZ instruction. If the lower 16-bits of #32FHex are not zeros, then assembler will convert MOVIZ to a MMOVIZ and a MMOVXI instruction.

**Example**

```
MMOVI32 MR1, #0x40400000 ; MR1 = 0x40400000
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR1, #0x4040

MMOVI32 MR2, #0x00000000 ; MR2 = 0x00000000
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR2, #0x0

MMOVI32 MR3, #0x40004001 ; MR3 = 0x40004001
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR3, #0x4000
                          ; MMOVXI MR3, #0x4001

MMOVI32 MR0, #0x00004040 ; MR0 = 0x00004040
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR0, #0x0000
                          ; MMOVXI MR0, #0x4040
```

**See also**

[MMOVIZ MRa, #16FHi](#)  
[MMOVXI MRa, #16FLoHex](#)  
[MMOVF32 MRa, #32F](#)

## MMOVIZ MRa, #16FHi *Load the Upper 16-Bits of a 32-Bit Floating-Point Register*

### Operands

MRa	floating-point register (MR0 to MR3)
#16FHi	A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0.

### Opcode

```
LSW: I I I I I I I I I I I I I I
MSW: 0 1 1 1 1 0 0 0 0 1 0 0 0 0 a a
```

### Description

Load the upper 16-bits of MRa with the immediate value #16FHi and clear the low 16-bits of MRa.

#16FHiHex is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. The assembler will only accept a decimal or hex immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

By itself, MMOVIZ is useful for loading a floating-point register with a constant in which the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). If a constant requires all 32-bits of a floating-point register to be initialized, then use MMOVIZ along with the MMOVXI instruction.

```
MRa(31:16) = #16FHi;
MRa(15:0) = 0;
```

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

### Pipeline

This is a single-cycle instruction.

### Example

```
; Load MR0 and MR1 with -1.5 (0xBFC00000)
MMOVIZ MR0, #0xBFC0 ; MR0 = 0xBFC00000 (1.5)
MMOVIZ MR1, #-1.5 ; MR1 = -1.5 (0xBFC00000)

; Load MR2 with pi = 3.141593 (0x40490FDB)
MMOVIZ MR2, #0x4049 ; MR2 = 0x40490000
MMOVXI MR2, #0x0FDB ; MR2 = 0x40490FDB
```

### See also

[MMOVF32 MRa, #32F](#)  
[MMOVIZ MRa, #32FHex](#)  
[MMOVXI MRa, #16FLoHex](#)

**MMOVZ16 MRa, mem16 *Load MRx With 16-bit Value***
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
mem16	16-bit source memory location

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 10aa addr
```

**Description**

Move the 16-bit value referenced by mem16 to the floating-point register indicated by MRa.

```
MRa(31:16) = 0;
MRa(15:0) = [mem16];
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified based on the integer results of the operation.

```
NF = 0;
if (MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**MMOVXI MRa, #16FLoHex** *Move Immediate to the Low 16-Bits of a Floating-Point Register*
**Operands**

MRa	CLA floating-point register (MR0 to MR3)
#16FLoHex	A 16-bit immediate hex value that represents the lower 16-bits of an IEEE 32-bit floating-point value. The upper 16-bits will not be modified.

**Opcode**

```
LSW: I I I I I I I I I I I I I I
MSW: 0 1 1 1 1 0 0 0 1 0 0 0 0 0 a a
```

**Description**

Load the low 16-bits of MRa with the immediate value #16FLoHex. #16FLoHex represents the lower 16-bits of an IEEE 32-bit floating-point value. The upper 16-bits of MRa will not be modified. MMOVXI can be combined with the MMOVIZ instruction to initialize all 32-bits of a MRa register.

```
MRa(15:0) = #16FLoHex;
MRa(31:16) = Unchanged;
```

**Flags**

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Load MR0 with pi = 3.141593 (0x40490FDB)
MMOVIZ    MR0, #0x4049    ; MR0 = 0x40490000
MMOVXI    MR0, #0x0FDB    ; MR0 = 0x40490FDB
```

**See also**

[MMOVIZ MRa, #16FHi](#)

**MMPYF32 MRa, MRb, MRc 32-Bit Floating-Point Multiply**
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)
MRc	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 0000 0000
```

**Description**

Multiply the contents of two floating-point registers.

```
MRa = MRb * MRc;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Calculate Num/Den using a Newton-Raphson algorithm for 1/Den
; Ye = Estimate(1/X)
; Ye = Ye*(2.0 - Ye*X)
; Ye = Ye*(2.0 - Ye*X)
;
_ClalTask1:
    MMOV32    MR1, @_Den    ; MR1 = Den
    MEINVF32  MR2, MR1     ; MR2 = Ye = Estimate(1/Den)
    MMPYF32   MR3, MR2, MR1 ; MR3 = Ye*Den
    MSUBF32   MR3, #2.0, MR3 ; MR3 = 2.0 - Ye*Den
    MMPYF32   MR2, MR2, MR3 ; MR2 = Ye = Ye*(2.0 - Ye*Den)
    MMPYF32   MR3, MR2, MR1 ; MR3 = Ye*Den
||  MMOV32    MR0, @_Num    ; MR0 = Num
    MSUBF32   MR3, #2.0, MR3 ; MR3 = 2.0 - Ye*Den
    MMPYF32   MR2, MR2, MR3 ; MR2 = Ye = Ye*(2.0 - Ye*Den)
||  MMOV32    MR1, @_Den    ; Reload Den To Set Sign
    MNEGF32   MR0, MR0, EQ  ; if(Den == 0.0) Change Sign Of Num
    MMPYF32   MR0, MR2, MR0 ; MR0 = Y = Ye*Num
    MMOV32    @_Dest, MR0   ; Store result
    MSTOP
```

**See also**

[MMPYF32 MRa, #16FHi, MRb](#)  
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRc, MRf](#)  
[MMPYF32 MRd, MRc, MRf || MMOV32 MRa, mem32](#)  
[MMPYF32 MRd, MRc, MRf || MMOV32 mem32, MRa](#)  
[MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRc, MRf](#)  
[MMACF32 MR3, MR2, MRd, MRc, MRf || MMOV32 MRa, mem32](#)



## MMPYF32 MRa, #16FHi, MRb 32-Bit Floating-Point Multiply

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
#16FHi	A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0.
MRb	CLA floating-point source register (MR0 to MR3)

### Opcode

LSW: I I I I I I I I I I I I I I I I I I  
MSW: 0111 0111 1000 bbaa

### Description

Multiply MRb with the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

MRa = MRb \* #16FHi:0;

This instruction can also be written as MMPYF32 MRa, MRb, #16FHi.

### Flags

This instruction modifies the following flags in the MSTF register:.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

### Pipeline

This is a single-cycle instruction.

### Example 1

```
; Same as example 2 but #16FHi is represented in float
MMOVIZ MR3, #2.0 ; MR3 = 2.0 (0x40000000)
MMPYF32 MR0, #3.0, MR3 ; MR0 = 3.0 * MR3 = 6.0 (0x40C00000)
MMOV32 @_X, MR0 ; Save the result in variable X
```

### Example 2

```
; Same as example 1 but #16FHi is represented in Hex
MMOVIZ MR3, #2.0 ; MR3 = 2.0 (0x40000000)
MMPYF32 MR0, #0x4040, MR3 ; MR0 = 0x4040 * MR3 = 6.0 (0x40C00000)
MMOV32 @_X, MR0 ; Save the result in variable X
```

**Example 3**

```

; Given X, M and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
;
_ClalTask2:
;
; Convert M, X and B from IQ24 to float
    MI32TOF32    MR0, @_M           ; MR0 = 0x4BC00000
    MI32TOF32    MR1, @_X           ; MR1 = 0x4C200000
    MI32TOF32    MR2, @_B           ; MR2 = 0xCB000000
    MMPYF32      MR0, MR0, #0x3380 ; M = 1/(1*2^24) * iqm = 1.5 (0x3FC00000)
    MMPYF32      MR1, MR1, #0x3380 ; X = 1/(1*2^24) * iqx = 2.5 (0x40200000)
    MMPYF32      MR2, MR2, #0x3380 ; B = 1/(1*2^24) * iqb = -.5 (0xBF000000)
    MMPYF32      MR3, MR0, MR1     ; M*X
    MADDF32      MR2, MR2, MR3     ; Y=MX+B = 3.25 (0x40500000)
; Convert Y from float32 to IQ24
    MMPYF32      MR2, MR2, #0x4B80 ; Y * 1*2^24
    MF32TOI32    MR2, MR2         ; IQ24(Y) = 0x03400000
    MMOV32 @_Y, MR2              ; store result
    MSTOP                          ; end of task

```

**See also**

[MMPYF32 MRa, MRb, #16FHi](#)  
[MMPYF32 MRa, MRb, MRc](#)  
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf](#)

## MMPYF32 MRa, MRb, #16FHi 32-Bit Floating-Point Multiply

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)
#16FHi	A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0.

### Opcode

```
LSW: I I I I I I I I I I I I I I I I I I
MSW: 0 1 1 1 0 1 1 1 1 0 0 0 b b a a
```

### Description

Multiply MRb with the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb * #16FHi:0;
```

This instruction can also be written as MMPYF32 MRa, #16FHi, MRb.

### Flags

This instruction modifies the following flags in the MSTF register:.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

### Pipeline

This is a single-cycle instruction.

### Example 1

```
;Same as example 2 but #16FHi is represented in float
MMOVIZ MR3, #2.0 ; MR3 = 2.0 (0x40000000)
MMPYF32 MR0, MR3, #3.0 ; MR0 = MR3 * 3.0 = 6.0 (0x40C00000)
MMOV32 @_X, MR0 ; Save the result in variable X
```

### Example 2

```
;Same as above example but #16FHi is represented in Hex
MMOVIZ MR3, #2.0 ; MR3 = 2.0 (0x40000000)
MMPYF32 MR0, MR3, #0x4040 ; MR0 = MR3 * 0x4040 = 6.0 (0x40C00000)
MMOV32 @_X, MR0 ; Save the result in variable X
```

**Example 3**

```

; Given X, M and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
_ClalTask2:
;
; Convert M, X and B from IQ24 to float
    MI32TOF32    MR0, @_M          ; MR0 = 0x4BC00000
    MI32TOF32    MR1, @_X          ; MR1 = 0x4C200000
    MI32TOF32    MR2, @_B          ; MR2 = 0xCB000000
    MMPYF32      MR0, #0x3380, MR0 ; M = 1/(1*2^24) * iqm = 1.5 (0x3FC00000)
    MMPYF32      MR1, #0x3380, MR1 ; X = 1/(1*2^24) * iqx = 2.5 (0x40200000)
    MMPYF32      MR2, #0x3380, MR2 ; B = 1/(1*2^24) * iqb = -.5 (0xBF000000)
    MMPYF32      MR3, MR0, MR1     ; M*X
    MADDF32      MR2, MR2, MR3     ; Y=MX+B = 3.25 (0x40500000)

; Convert Y from float32 to IQ24
    MMPYF32      MR2, #0x4B80, MR2 ; Y * 1*2^24
    MF32TOI32    MR2, MR2         ; IQ24(Y) = 0x03400000
    MMOV32       @_Y, MR2         ; store result
    MSTOP
; end of task

```

**See also**

[MMPYF32 MRa, #16FHi, MRb](#)  
[MMPYF32 MRa, MRb, MRc](#)

## MMPYF32 MRa, MRb, MRc||MADDF32 MRd, MRe, MRf 32-Bit Floating-Point Multiply with Parallel Add

### Operands

MRa	CLA floating-point destination register for MMPYF32 (MR0 to MR3) MRa cannot be the same register as MRd
MRb	CLA floating-point source register for MMPYF32 (MR0 to MR3)
MRc	CLA floating-point source register for MMPYF32 (MR0 to MR3)
MRd	CLA floating-point destination register for MADDF32 (MR0 to MR3) MRd cannot be the same register as MRa
MRe	CLA floating-point source register for MADDF32 (MR0 to MR3)
MRf	CLA floating-point source register for MADDF32 (MR0 to MR3)

### Opcode

LSW: 0000 ffee ddcc bbaa  
MSW: 0111 1010 0000 0000

### Description

Multiply the contents of two floating-point registers with parallel addition of two registers.

MRa = MRb \* MRc;  
MRd = MRe + MRf;

### Restrictions

The destination register for the MMPYF32 and the MADDF32 must be unique. That is, MRa cannot be the same register as MRd.

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 or MADDF32 generates an underflow condition.
- LVF = 1 if MMPYF32 or MADDF32 generates an overflow condition.

### Pipeline

Both MMPYF32 and MADDF32 complete in a single cycle.

**Example**

```

; Perform 5 multiply and accumulate operations:
;
; X and Y are 32-bit floating point arrays
;
; 1st multiply: A = X0 * Y0
; 2nd multiply: B = X1 * Y1
; 3rd multiply: C = X2 * Y2
; 4th multiply: D = X3 * Y3
; 5th multiply: E = X3 * Y3
;
; Result = A + B + C + D + E
;
_ClalTask1:
    MMOV16    MAR0, #_X          ; MAR0 points to X array
    MMOV16    MAR1, #_Y          ; MAR1 points to Y array
    MNOOP     ; Delay for MAR0, MAR1 load
    MNOOP     ; Delay for MAR0, MAR1 load
    ; <-- MAR0 valid
    MMOV32    MR0, *MAR0[2]++    ; MR0 = X0, MAR0 += 2
    ; <-- MAR1 valid
    MMOV32    MR1, *MAR1[2]++    ; MR1 = Y0, MAR1 += 2

    MMPYF32   MR2, MR0, MR1      ; MR2 = A = X0 * Y0
    || MMOV32  MR0, *MAR0[2]++    ; In parallel MR0 = X1, MAR0 += 2
    MMOV32    MR1, *MAR1[2]++    ; MR1 = Y1, MAR1 += 2

    MMPYF32   MR3, MR0, MR1      ; MR3 = B = X1 * Y1
    || MMOV32  MR0, *MAR0[2]++    ; In parallel MR0 = X2, MAR0 += 2
    MMOV32    MR1, *MAR1[2]++    ; MR1 = Y2, MAR2 += 2

    MMACF32   MR3, MR2, MR2, MR0, MR1 ; MR3 = A + B, MR2 = C = X2 * Y2
    || MMOV32  MR0, *MAR0[2]++    ; In parallel MR0 = X3
    MMOV32    MR1, *MAR1[2]++    ; MR1 = Y3

    MMACF32   MR3, MR2, MR2, MR0, MR1 ; MR3 = (A + B) + C, MR2 = D = X3 * Y3
    || MMOV32  MR0, *MAR0
    MMOV32    MR1, *MAR1          ; MR1 = Y4

    MMPYF32   MR2, MR0, MR1      ; MR2 = E = X4 * Y4
    || MADD32  MR3, MR3, MR2      ; in parallel MR3 = (A + B + C) + D

    MADD32    MR3, MR3, MR2      ; MR3 = (A + B + C + D) + E
    MMOV32    @_Result, MR3      ; Store the result
    MSTOP     ; end of task

```

**See also**
[MMACF32 MR3, MR2, MRd, MRe, MRf || MMOV32 MRa, mem32](#)

## MMPYF32 MRd, MRe, MRf ||MMOV32 MRa, mem32 32-Bit Floating-Point Multiply with Parallel Move

### Operands

MRd	CLA floating-point destination register for the MMPYF32 (MR0 to MR3) MRd cannot be the same register as MRa
MRe	CLA floating-point source register for the MMPYF32 (MR0 to MR3)
MRf	CLA floating-point source register for the MMPYF32 (MR0 to MR3)
MRa	CLA floating-point destination register for the MMOV32 (MR0 to MR3) MRa cannot be the same register as MRd
mem32	32-bit memory location accessed using one of the available addressing modes. This will be the source of the MMOV32.

### Opcode

LSW: mmmm mmmm mmmm mmmm  
MSW: 0000 ffee ddaa addr

### Description

Multiply the contents of two floating-point registers and load another.

MRd = MRe \* MRf;  
MRa = [mem32];

### Restrictions

The destination register for the MMPYF32 and the MMOV32 must be unique. That is, MRa cannot be the same register as MRd.

### Flags

This instruction modifies the following flags in the MSTF register:.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

The MMOV32 Instruction will set the NF and ZF flags as follows:

NF = MRa(31);  
ZF = 0;  
if(MRa(30:23) == 0) { ZF = 1; NF = 0; }

### Pipeline

Both MMPYF32 and MMOV32 complete in a single cycle.

### Example 1

```
; Given M1, X1 and B1 are 32-bit floating point
; Calculate Y1 = M1*X1+B1
;
_Cla1Task1:
    MMOV32    MR0, @M1        ; Load MR0 with M1
    MMOV32    MR1, @X1        ; Load MR1 with X1
    MMPYF32   MR1, MR1, MR0    ; Multiply M1*X1
    || MMOV32  MR0, @B1        ; and in parallel load MR0 with B1
    MADDF32   MR1, MR1, MR0    ; Add M*X1 to B1 and store in MR1
    MMOV32    @Y1, MR1        ; Store the result
    MSTOP
```

**Example 2**

```

; Given A, B and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A * B) * C
;
_ClalTask2:
    MMOV32    MR0, @A        ; Load MR0 with A
    MMOV32    MR1, @B        ; Load MR1 with B
    MMPYF32   MR1, MR1, MR0  ; Multiply A*B
|| MMOV32    MR0, @C        ; and in parallel load MR0 with C
    MMPYF32   MR1, MR1, MR0  ; Multiply (A*B) by C
|| MMOV32    @Y2, MR1       ; and in parallel store A*B
    MMOV32    @Y3, MR1       ; Store the result
    MSTOP                    ; end of task

```

**See also**

[MMPYF32 MRd, MRe, MRf || MMOV32 mem32, MRa](#)  
[MMACF32 MR3, MR2, MRd, MRe, MRf || MMOV32 MRa, mem32](#)



## MMPYF32 MRd, MRe, MRf ||MMOV32 mem32, MRa 32-Bit Floating-Point Multiply with Parallel Move

### Operands

MRd	CLA floating-point destination register for the MMPYF32 (MR0 to MR3)
MRe	CLA floating-point source register for the MMPYF32 (MR0 to MR3)
MRf	CLA floating-point source register for the MMPYF32 (MR0 to MR3)
mem32	32-bit memory location accessed using one of the available addressing modes. This will be the destination of the MMOV32.
MRa	CLA floating-point source register for the MMOV32 (MR0 to MR3)

### Opcode

LSW: mmmm mmmm mmmm mmmm  
MSW: 0100 ffee ddaa addr

### Description

Multiply the contents of two floating-point registers and move from memory to register.

MRd = MRe \* MRf;  
[mem32] = MRa;

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

### Pipeline

MMPYF32 and MMOV32 both complete in a single cycle.

### Example

```
; Given A, B and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A * B) * C
;
_ClalTask2:
    MMOV32    MR0, @A           ; Load MR0 with A
    MMOV32    MR1, @B           ; Load MR1 with B
    MMPYF32   MR1, MR1, MR0     ; Multiply A*B
||    MMOV32    MR0, @C           ; and in parallel load MR0 with C
    MMPYF32   MR1, MR1, MR0     ; Multiply (A*B) by C
||    MMOV32    @Y2, MR1         ; and in parallel store A*B
    MMOV32    @Y3, MR1         ; Store the result
    MSTOP
```

### See also

[MMPYF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)  
[MMACF32 MR3, MR2, MRd, MRe, MRf || MMOV32 MRa, mem32](#)

**MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf 32-Bit Floating-Point Multiply with Parallel Subtract**
**Operands**

MRa	CLA floating-point destination register for MMPYF32 (MR0 to MR3) MRa cannot be the same register as MRd
MRb	CLA floating-point source register for MMPYF32 (MR0 to MR3)
MRc	CLA floating-point source register for MMPYF32 (MR0 to MR3)
MRd	CLA floating-point destination register for MSUBF32 (MR0 to MR3) MRd cannot be the same register as MRa
MRe	CLA floating-point source register for MSUBF32 (MR0 to MR3)
MRf	CLA floating-point source register for MSUBF32 (MR0 to MR3)

**Opcode**

```
LSW: 0000 ffee ddc bbaa
MSW: 0111 1010 0100 0000
```

**Description**

Multiply the contents of two floating-point registers with parallel subtraction of two registers.

```
MRa = MRb * MRc;
MRd = MRe - MRf;
```

**Restrictions**

The destination register for the MMPYF32 and the MSUBF32 must be unique. That is, MRa cannot be the same register as MRd.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 or MSUBF32 generates an underflow condition.
- LVF = 1 if MMPYF32 or MSUBF32 generates an overflow condition.

**Pipeline**

MMPYF32 and MSUBF32 both complete in a single cycle.

**Example**

```
; Given A, B and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A - B)
;
_ClalTask2:
    MOV32    MR0, @A           ; Load MR0 with A
    MOV32    MR1, @B           ; Load MR1 with B
    MMPYF32  MR2, MR0, MR1     ; Multiply (A*B)
    || MSUBF32 MR3, MR0, MR1   ; and in parallel Sub (A-B)
    MOV32    @Y2, MR2          ; Store A*B
    MOV32    @Y3, MR3          ; Store A-B
    MSTOP                                     ; end of task
```

**See also**

[MSUBF32 MRa, MRb, MRc](#)  
[MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)  
[MSUBF32 MRd, MRe, MRf || MMOV32 mem32, MRa](#)

## MNEGF32 MRa, MRb{, CNDF} *Conditional Negation*

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)
CNDF	condition tested

### Opcode

```
LSW: 0000 0000 cndf bbaa
MSW: 0111 1010 1000 0000
```

### Description

```
if (CNDF == true) {MRa = - MRb; }
else {MRa = MRb; }
```

CNDF is one of the following conditions:

Encode <sup>(5)</sup>	CNDF	Description	MSTF Flags Tested
0000	NEQ	Not equal to zero	ZF == 0
0001	EQ	Equal to zero	ZF == 1
0010	GT	Greater than zero	ZF == 0 AND NF == 0
0011	GEQ	Greater than or equal to zero	NF == 0
0100	LT	Less than zero	NF == 1
0101	LEQ	Less than or equal to zero	ZF == 1 OR NF == 1
1010	TF	Test flag set	TF == 1
1011	NTF	Test flag not set	TF == 0
1100	LU	Latched underflow	LUF == 1
1101	LV	Latched overflow	LVF == 1
1110	UNC	Unconditional	None
1111	UNCF <sup>(6)</sup>	Unconditional with flag modification	None

<sup>(5)</sup> Values not shown are reserved.

<sup>(6)</sup> This is the default operation if no CNDF field is specified. This condition will allow the ZF, and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

### Pipeline

This is a single-cycle instruction.

### Example 1

```
; Show the basic operation of MNEGF32
;
MMOVIZ MR0, #5.0 ; MR0 = 5.0 (0x40A00000)
MMOVIZ MR1, #4.0 ; MR1 = 4.0 (0x40800000)
MMOVIZ MR2, #-1.5 ; MR2 = -1.5 (0xBF000000)
MMPYF32 MR3, MR1, MR2 ; MR3 = -6.0
MMPYF32 MR0, MR0, MR1 ; MR0 = 20.0
MMOVIZ MR1, #0.0
MCMPPF32 MR3, MR1 ; NF = 1
MNEGF32 MR3, MR3, LT ; if NF = 1, MR3 = 6.0
MCMPPF32 MR0, MR1 ; NF = 0
MNEGF32 MR0, MR0, GEQ ; if NF = 0, MR0 = -20.0
```

**Example 2**

```

; Calculate Num/Den using a Newton-Raphson algorithm for 1/Den
; Ye = Estimate(1/X)
; Ye = Ye*(2.0 - Ye*X)
; Ye = Ye*(2.0 - Ye*X)
;
_ClalTask1:
    MMOV32    MR1, @_Den        ; MR1 = Den
    MEINVF32  MR2, MR1         ; MR2 = Ye = Estimate(1/Den)
    MMPYF32   MR3, MR2, MR1    ; MR3 = Ye*Den
    MSUBF32   MR3, #2.0, MR3   ; MR3 = 2.0 - Ye*Den
    MMPYF32   MR2, MR2, MR3    ; MR2 = Ye = Ye*(2.0 - Ye*Den)
    MMPYF32   MR3, MR2, MR1    ; MR3 = Ye*Den
| | MMOV32    MR0, @_Num        ; MR0 = Num
| | MSUBF32   MR3, #2.0, MR3   ; MR3 = 2.0 - Ye*Den
| | MMPYF32   MR2, MR2, MR3    ; MR2 = Ye = Ye*(2.0 - Ye*Den)
| | MMOV32    MR1, @_Den        ; Reload Den To Set Sign
| | MNEGF32   MR0, MR0, EQ     ; if(Den == 0.0) Change Sign Of Num
| | MMPYF32   MR0, MR2, MR0    ; MR0 = Y = Ye*Num
| | MMOV32    @_Dest, MR0      ; Store result
| | MSTOP
; end of task

```

**See also**
[MABSF32 MRa, MRb](#)

## MNOP *No Operation*

### Operands

none This instruction does not have any operands

### Opcode

LSW: 0000 0000 0000 0000  
MSW: 0111 1111 1010 0000

### Description

Do nothing. This instruction is used to fill required pipeline delay slots when other instructions are not available to fill the slots.

### Flags

This instruction does not modify flags in the MSTF register.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

### Pipeline

This is a single-cycle instruction.

### Example

```

; X is an array of 32-bit floating-point values
; Find the maximum value in an array X
; and store it in Result
;
_ClalTask1:
    MMOVI16    MAR1, #_X           ; Start address
    MUII16TOF32 MR0, @_len        ; Length of the array
    MNOP                          ; delay for MAR1 load
    MNOP                          ; delay for MAR1 load
    MMOV32     MR1, *MAR1[2]++    ; MR1 = X0
LOOP
    MMOV32     MR2, *MAR1[2]++    ; MR2 = next element
    MMAXF32    MR1, MR2           ; MR1 = MAX(MR1, MR2)
    MADDF32    MR0, MR0, #-1.0    ; Decrement the counter
    MCMFP32    MR0 #0.0          ; Set/clear flags for MBCNDD
    MNOP                          ; Too late to affect MBCNDD
    MNOP                          ; Too late to affect MBCNDD
    MNOP                          ; Too late to affect MBCNDD
    MBCNDD     LOOP, NEQ         ; Branch if not equal to zero
    MMOV32     @_Result, MR1     ; Always executed
    MNOP                          ; Pad to separate MBCNDD and MSTOP
    MNOP                          ; Pad to separate MBCNDD and MSTOP
    MSTOP                                           ; End of task

```

### See also

**MOR32 MRa, MRb, MRc *Bitwise OR***
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)
MRc	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 1000 0000
```

**Description**

Bitwise OR of MRb with MRc.

```
MARa(31:0) = MARb(31:0) OR MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ MR0, #0x5555 ; MR0 = 0x5555AAAA
MMOVXI MR0, #0xAAAA

MMOVIZ MR1, #0x5432 ; MR1 = 0x5432FEDC
MMOVXI MR1, #0xFEDC

; 0101 OR 0101 = 0101 (5)
; 0101 OR 0100 = 0101 (5)
; 0101 OR 0011 = 0111 (7)
; 0101 OR 0010 = 0111 (7)
; 1010 OR 1111 = 1111 (F)
; 1010 OR 1110 = 1110 (E)
; 1010 OR 1101 = 1111 (F)
; 1010 OR 1100 = 1110 (E)

MOR32 MR2, MR1, MR0 ; MR3 = 0x5555FEFE
```

**See also**

[MAND32 MRa, MRb, MRc](#)  
[MXOR32 MRa, MRb, MRc](#)

**MRCNDD {CNDF}     *Return Conditional Delayed***
**Operands**

CNDF	optional condition.
------	---------------------

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1001 1010 cndf
```

**Description**

If the specified condition is true, then the RPC field of MSTF is loaded into MPC and fetching continues from that location. Otherwise program fetches will continue without the return.

Please refer to the pipeline section for important information regarding this instruction.

```
if (CNDF == TRUE) MPC = RPC;
```

CNDF is one of the following conditions:

Encode <sup>(7)</sup>	CNDF	Description	MSTF Flags Tested
0000	NEQ	Not equal to zero	ZF == 0
0001	EQ	Equal to zero	ZF == 1
0010	GT	Greater than zero	ZF == 0 AND NF == 0
0011	GEQ	Greater than or equal to zero	NF == 0
0100	LT	Less than zero	NF == 1
0101	LEQ	Less than or equal to zero	ZF == 1 OR NF == 1
1010	TF	Test flag set	TF == 1
1011	NTF	Test flag not set	TF == 0
1100	LU	Latched underflow	LUF == 1
1101	LV	Latched overflow	LVF == 1
1110	UNC	Unconditional	None
1111	UNCF <sup>(8)</sup>	Unconditional with flag modification	None

<sup>(7)</sup> Values not shown are reserved.

<sup>(8)</sup> This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction does not modify flags in the MSTF register.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

The MRCNDD instruction by itself is a single-cycle instruction. As shown in [Table 5-20](#), for each return 6 instruction slots are executed; three before the return instruction (d5-d7) and three after the return instruction (d8-d10). The total number of cycles for a return taken or not taken depends on the usage of these slots. That is, the number of cycles depends on how many slots are filled with a MNOP as well as which slots are filled. The effective number of cycles for a return can, therefore, range from 1 to 7 cycles. The number of cycles for a return taken may not be the same as for a return not taken.

Referring to the following code fragment and the pipeline diagrams in [Table 5-20](#) and [Table 5-21](#), the instructions before and after MRCNDD have the following properties:

```

;
;
<Instruction 1>    ; I1 Last instruction that can affect flags for
                  ; the MCCNDD operation
<Instruction 2>    ; I2 Cannot be stop, branch, call or return
<Instruction 3>    ; I3 Cannot be stop, branch, call or return
<Instruction 4>    ; I4 Cannot be stop, branch, call or return

MCCNDD _func, NEQ ; Call to func if not equal to zero
                  ; Three instructions after MCCNDD are always
                  ; executed whether the call is taken or not
<Instruction 5>    ; I5 Cannot be stop, branch, call or return
<Instruction 6>    ; I6 Cannot be stop, branch, call or return
<Instruction 7>    ; I7 Cannot be stop, branch, call or return
<Instruction 8>    ; I8 The address of this instruction is saved
                  ; in the RPC field of the MSTF register.
                  ; Upon return this value is loaded into MPC
                  ; and fetching continues from this point.

<Instruction 9>    ; I9
<Instruction 10>   ; I10
....
....
_func:
<Destination 1>   ; d1 Can be any instruction
<Destination 2>   ; d2
<Destination 3>   ; d3
<Destination 4>   ; d4 Last instruction that can affect flags for
                  ; the MRCNDD operation
<Destination 5>   ; d5 Cannot be stop, branch, call or return
<Destination 6>   ; d6 Cannot be stop, branch, call or return
<Destination 7>   ; d7 Cannot be stop, branch, call or return

MRCNDD NEQ        ; Return to <Instruction 8> if not equal to zero
                  ; Three instructions after MRCNDD are always
                  ; executed whether the return is taken or not
<Destination 8>   ; d8 Cannot be stop, branch, call or return
<Destination 9>   ; d9 Cannot be stop, branch, call or return
<Destination 10>  ; d10 Cannot be stop, branch, call or return
<Destination 11>  ; d11
<Destination 12>  ; d12
....
....
MSTOP
....

```

- **d4**

- d4 is the last instruction that can effect the CNDF flags for the MRCNDD instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to return or not when MRCNDD is in the D2 phase.
- There are no restrictions on the type of instruction for d4.

- **d5, d6 and d7**

- The three instructions proceeding MRCNDD can change MSTF flags but will have no effect on whether the MRCNDD instruction makes the return or not. This is



because the flag modification will occur after the D2 phase of the MRCNDD instruction.

- These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.
- **d8, d9 and d10**
  - The three instructions following MRCNDD are always executed irrespective of whether the return is taken or not.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.

**Table 5-20. Pipeline Activity For MRCNDD, Return Not Taken**

Instruction	F1	F2	D1	D2	R1	R2	E	W
d4	d4	d3	d2	d1	l7	l6	l5	
d5	d5	d4	d3	d2	d1	l7	l6	
d6	d6	d5	d4	d3	d2	d1	i7	
d7	d7	d6	d5	d4	d3	d2	d1	
MRCNDD	MRCNDD	d7	d6	d5	d4	d3	d2	
d8	d8	MRCNDD	d7	d6	d5	d4	d3	
d9	d9	d8	MRCNDD	d7	d6	d5	d4	
d10	d10	d9	d8	MRCNDD	d7	d6	d5	
d11	d11	d10	d9	d8	-	d7	d6	
d12	d12	d11	d10	d9	d8	-	d7	
etc....	....	d12	d11	d10	d9	d8	-	
....	....	....	d12	d11	d10	d9	d8	
....	....	....	....	d12	d11	d10	d9	
					d12	d11	d10	
						d12	d11	
							d12	

**Table 5-21. Pipeline Activity For MRCNDD, Return Taken**

Instruction	F1	F2	D1	D2	R1	R2	E	W
d4	d4	d3	d2	d1	l7	l6	l5	
d5	d5	d4	d3	d2	d1	l7	l6	
d6	d6	d5	d4	d3	d2	d1	i7	
d7	d7	d6	d5	d4	d3	d2	d1	
MRCNDD	MRCNDD	d7	d6	d5	d4	d3	d2	
d8	d8	MRCNDD	d7	d6	d5	d4	d3	
d9	d9	d8	MRCNDD	d7	d6	d5	d4	
d10	d10	d9	d8	MRCNDD	d7	d6	d5	
l8	l8	d10	d9	d8	-	d7	d6	
l9	l9	l8	d10	d9	d8	-	d7	
l10	l10	l9	l8	d10	d9	d8	-	
etc....	....	l10	l9	l8	d10	d9	d8	
....	....	....	l10	l9	l8	d10	d9	
....	....	....	....	l10	l9	l8	d10	
					l10	l9	l8	
						l10	l9	
							l10	

**Example**

;

**See also**

[MBCNDD #16BitDest, CNDF](#)  
[MCCNDD 16BitDest, CNDF](#)  
[MMOV32 mem32, MSTF](#)  
[MMOV32 MSTF, mem32](#)

**MSETC BGINTM**      *Set Background Task Interrupt Mask*
**Operands**

none                      This instruction does not have any operands

**Opcode**

LSW: 0000 0000 0000 0000  
MSW: 0111 1111 0101 0000

**Description**

This instruction will set the background task interrupt mask (BGINTM) bit in the MSTSBGRND register, making any code thereafter un-interruptible. No other higher priority task will be able to interrupt the background task until the BGINTM is cleared. This instruction sets the BGINTM bit at the end of its D2 phase.

Note: This instruction does not require the MEALLOW bit to be asserted before, or de-asserted after, setting BGINTM.

**Flags**

This instruction does not modify the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MSETC BGINTM                      ; Set the MSTSBGRND.BGINTM bit
                                    ; to prevent any other tasks from
                                    ; interrupting the background task
```

**See also**

[MCLRC BGINTM](#)

**MSETFLG FLAG, VALUE *Set or Clear Selected Floating-Point Status Flags***
**Operands**

FLAG	8 bit mask indicating which floating-point status flags to change.
VALUE	8 bit mask indicating the flag value; 0 or 1.

**Opcode**

```
LSW: FFFF FFFF VVVV VVVV
MSW: 0111 1001 1100 0000
```

**Description**

The MSETFLG instruction is used to set or clear selected floating-point status flags in the MSTF register. The FLAG field is an 11-bit value that indicates which flags will be changed. That is, if a FLAG bit is set to 1 it indicates that flag will be changed; all other flags will not be modified. The bit mapping of the FLAG field is shown below:

RNDF3 2	reserve d	reserve d	TF	reserve d	reserved	ZF	NF	LUF	LVF
9	8	7	6	5	4	3	2	1	0

The VALUE field indicates the value the flag should be set to; 0 or 1.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	Yes	Yes	Yes	Yes	Yes

Any flag can be modified by this instruction. The MEALLOW and RPC fields cannot be modified with this instruction.

**Pipeline**

This is a single-cycle instruction.

**Example**

To make it easier and legible, the assembler accepts a FLAG=VALUE syntax for the MSETFLG operation as shown below:

```
MSETFLG RNDF32=0, TF=0, NF=1; FLAG = 11000100; VALUE = 00XXX1XX;
```

**See also**

[MMOV32 mem32, MSTF](#)  
[MMOV32 MSTF, mem32](#)

**MSTOP****Stop Task****Operands**

none	This instruction does not have any operands
------	---------------------------------------------

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1000 0000
```

**Description**

The MSTOP instruction must be placed to indicate the end of each task. In addition, placing MSTOP in unused memory locations within the CLA program RAM can be useful for debugging and preventing run away CLA code. When MSTOP enters the D2 phase of the pipeline, the MIRUN flag for the task is cleared and the associated interrupt is flagged in the PIE vector table.

There are three special cases that can occur when single-stepping a task such that the MPC reaches the MSTOP instruction.

1. If you are single-stepping or halted in "task A" and "task B" comes in before the MPC reaches the MSTOP, then "task B" will start if you continue to step through the MSTOP instruction. Basically if "task B" is pending before the MPC reaches MSTOP in "task A" then there is no issue in "task B" starting and no special action is required.
2. In this case you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. If "task B" comes in at this point, it will be flagged in the MIFR register but it may or may not start if you continue to single-step through the MSTOP instruction of "task A". It depends on exactly when the new task comes in. To reliably start "task B" perform a soft reset and reconfigure the MIER bits. Once this is done, you can start single-stepping "task B".
3. Case 2 can be handled slightly differently if there is control over when "task B" comes in (for example using the IACK instruction to start the task). In this case you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. Before forcing "task B", run free to force the CLA out of the debug state. Once this is done you can force "task B" and continue debugging.

**Restrictions**

The MSTOP instruction cannot be placed 3 instructions before or after a [MBCNDD](#), [MCCNDD](#) or [MRCNDD](#) instruction.

**Flags**

This instruction does not modify flags in the MSTF register.

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction. [Table 5-22](#) shows the pipeline behavior of the MSTOP instruction. The MSTOP instruction cannot be placed with 3 instructions of a [MBCNDD](#), [MCCNDD](#) or [MRCNDD](#) instruction.

**Table 5-22. Pipeline Activity For MSTOP**

Instruction	F1	F2	D1	D2	R1	R2	E	W
I1	I1							
I2	I2	I1						
I3	I3	I2	I1					
MSTOP	MSTOP	I3	I2	I1				
I4	I4	MSTOP	I3	I2	I1			
I5	I5	I4	MSTOP	I3	I2	I1		
I6	I6	I5	I4	MSTOP	I3	I2	I1	
New Task Arbitrated and Piroitized	-	-	-	-	-	I3	I2	
New Task Arbitrated and Piroitized	-	-	-	-	-	-	I3	
I1	I1	-	-	-	-	-	-	-
I2	I2	I1	-	-	-	-	-	-
I3	I3	I2	I1	-	-	-	-	-
I4	I4	I3	I2	I1	-	-	-	-
I5	I5	I4	I3	I2	I1	-	-	-
I6	I6	I5	I4	I3	I2	I1	-	-
I7	I7	I6	I5	I4	I3	I2	I1	
etc ....								

**Example**

```

; Given A = (int32)1
;       B = (int32)2
;       C = (int32)-7
;
; Calculate Y2 = A - B - C
_C1a1Task3:
    MMOV32  MR0, @_A      ; MR0 = 1 (0x00000001)
    MMOV32  MR1, @_B      ; MR1 = 2 (0x00000002)
    MMOV32  MR2, @_C      ; MR2 = -7 (0xFFFFFFFF9)
    MSUB32  MR3, MR0, MR1 ; A + B
    MSUB32  MR3, MR3, MR2 ; A + B + C = 6 (0x00000006)
    MMOV32  @_y2, MR3     ; Store y2
    MSTOP                    ; End of task

```

**See also**
[MDEBUGSTOP](#) , [MDEBUGSTOP1](#)

## MSUB32 MRa, MRb, MRc 32-Bit Integer Subtraction

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point destination register (MR0 to MR3)
MRc	CLA floating-point destination register (MR0 to MR3)

### Opcode

LSW: 0000 0000 00cc bbaa  
MSW: 0111 1100 1110 0000

### Description

32-bit integer addition of MRb and MRc.

$MARa(31:0) = MARb(31:0) - MRc(31:0);$

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

### Pipeline

This is a single-cycle instruction.

### Example

```
; Given A = (int32)1
;      B = (int32)2
;      C = (int32)-7
;
;
Calculate Y2 = A - B - C
;
_ClalTask3:
    MMOV32 MR0, @_A      ; MR0 = 1 (0x00000001)
    MMOV32 MR1, @_B      ; MR1 = 2 (0x00000002)
    MMOV32 MR2, @_C      ; MR2 = -7 (0xFFFFFFFF9)
    MSUB32 MR3, MR0, MR1 ; A + B
    MSUB32 MR3, MR3, MR2 ; A + B + C = 6 (0x00000006)
    MMOV32 @_y2, MR3     ; Store y2
    MSTOP                ; End of task
```

### See also

[MADD32 MRa, MRb, MRc](#)  
[MAND32 MRa, MRb, MRc](#)  
[MASR32 MRa, #SHIFT](#)  
[MLSL32 MRa, #SHIFT](#)  
[MLSR32 MRa, #SHIFT](#)  
[MOR32 MRa, MRb, MRc](#)  
[MXOR32 MRa, MRb, MRc](#)

**MSUBF32 MRa, MRb, MRc 32-Bit Floating-Point Subtraction**
**Operands**

MRa	CLA floating-point destination register (MR0 to R1)
MRb	CLA floating-point source register (MR0 to R1)
MRc	CLA floating-point source register (MR0 to R1)

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 0100 0000
```

**Description**

Subtract the contents of two floating-point registers

$$MRa = MRb - MRc;$$
**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given A, B and C are 32-bit floating-point numbers
; Calculate Y2 = A + B - C
;
_ClalTask5:
    MMOV32    MR0, @_A      ; Load MR0 with A
    MMOV32    MR1, @_B      ; Load MR1 with B
    MADDF32   MR0, MR1, MR0 ; Add A + B
|| MMOV32    MR1, @_C      ; and in parallel load C
    MSUBF32   MR0, MR0, MR1 ; Subtract C from (A + B)
    MMOV32    @Y, MR0      ; (A+B) - C
    MSTOP
```

**See also**

[MSUBF32 MRa, #16FHi, MRb](#)  
[MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)  
[MSUBF32 MRd, MRe, MRf || MMOV32 mem32, MRa](#)  
[MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf](#)



**MSUBF32 MRa, #16FHi, MRb 32-Bit Floating-Point Subtraction**
**Operands**

MRa	CLA floating-point destination register (MR0 to R1)
#16FHi	A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0.
MRb	CLA floating-point source register (MR0 to R1)

**Opcode**

```
LSW: I I I I I I I I I I I I I I I I
MSW: 0111 1000 0000 baaa
```

**Description**

Subtract MRb from the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = #16FHi:0 - MRb;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Y = sqrt(X)
; Ye = Estimate(1/sqrt(X));
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Y = X*Ye
;
_ClalTask3:
    MMOV32    MR0, @_x           ; MR0 = X
    MEISQRTF32 MR1, MR0         ; MR1 = Ye = Estimate(1/sqrt(X))
    MMOV32    MR1, @_x, EQ       ; if(X == 0.0) Ye = 0.0
    MMPYF32   MR3, MR0, #0.5     ; MR3 = X*0.5
    MMPYF32   MR2, MR1, MR3     ; MR2 = Ye*X*0.5
    MMPYF32   MR2, MR1, MR2     ; MR2 = Ye*Ye*X*0.5
    MSUBF32   MR2, #1.5, MR2     ; MR2 = 1.5 - Ye*Ye*X*0.5
    MMPYF32   MR1, MR1, MR2     ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    MMPYF32   MR2, MR1, MR3     ; MR2 = Ye*X*0.5
    MMPYF32   MR2, MR1, MR2     ; MR2 = Ye*Ye*X*0.5
    MSUBF32   MR2, #1.5, MR2     ; MR2 = 1.5 - Ye*Ye*X*0.5
    MMPYF32   MR1, MR1, MR2     ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    MMPYF32   MR0, MR1, MR0     ; MR0 = Y = Ye*X
    MMOV32    @_y, MR0          ; Store Y = sqrt(X)
    MSTOP
```

**See also**

[MSUBF32 MRa, MRb, MRc](#)  
[MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)  
[MSUBF32 MRd, MRe, MRf || MMOV32 mem32, MRa](#)  
[MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf](#)

**MSUBF32 MRd, MRe, MRf ||MMOV32 MRa, mem32 32-Bit Floating-Point Subtraction with Parallel Move**
**Operands**

MRd	CLA floating-point destination register (MR0 to MR3) for the MSUBF32 operation MRd cannot be the same register as MRa
MRe	CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation
MRf	CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation
MRa	CLA floating-point destination register (MR0 to MR3) for the MMOV32 operation MRa cannot be the same register as MRd
mem32	32-bit memory location accessed using one of the available addressing modes. Source for the MMOV32 operation.

**Opcode**

```
LSW: rrrrrr rrrrrr rrrrrr rrrrrr
MSW: 0010 ffee ddaa addr
```

**Description**

Subtract the contents of two floating-point registers and move from memory to a floating-point register.

```
MRd = MRe - MRf;
MRa = [mem32];
```

**Restrictions**

The destination register for the MSUBF32 and the MMOV32 must be unique. That is, MRa cannot be the same register as MRd.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

The MMOV32 Instruction will set the NF and ZF flags as follows:

**Pipeline**

Both MSUBF32 and MMOV32 complete in a single cycle.

**Example**

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; }
```

**See also**

[MSUBF32 MRa, MRb, MRc](#)  
[MSUBF32 MRa, #16FHi, MRb](#)  
[MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf](#)

## MSUBF32 MRd, MRe, MRf ||MMOV32 mem32, MRa *32-Bit Floating-Point Subtraction with Parallel Move*

### Operands

MRd	CLA floating-point destination register (MR0 to MR3) for the MSUBF32 operation
MRe	CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation
MRf	CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation
mem32	32-bit destination memory location for the MMOV32 operation
MRa	CLA floating-point source register (MR0 to MR3) for the MMOV32 operation

### Opcode

LSW: mmmm mmmm mmmm mmmm  
MSW: 0110 ffee ddaa addr

### Description

Subtract the contents of two floating-point registers and move from a floating-point register to memory.

MRd = MRe - MRf;  
[mem32] = MRa;

### Flags

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	Yes	Yes

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

### Pipeline

Both MSUBF32 and MMOV32 complete in a single cycle.

### Example

### See also

[MSUBF32 MRa, MRb, MRc](#)  
[MSUBF32 MRa, #16FHi, MRb](#)  
[MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)  
[MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf](#)

**MSWAPF MRa, MRb {, CNDF} Conditional Swap**
**Operands**

MRa	CLA floating-point register (MR0 to MR3)
MRb	CLA floating-point register (MR0 to MR3)
CNDF	Optional condition tested based on the MSTF flags

**Opcode**

```
LSW: 0000 0000 CNDF bbaa
MSW: 0111 1011 0000 0000
```

**Description**

Conditional swap of MRa and MRb.

```
if (CNDF == true) swap MRa and MRb;
```

CNDF is one of the following conditions:

Encode <sup>(1)</sup>	CNDF	Description	MSTF Flags Tested
0000	NEQ	Not equal to zero	ZF == 0
0001	EQ	Equal to zero	ZF == 1
0010	GT	Greater than zero	ZF == 0 AND NF == 0
0011	GEQ	Greater than or equal to zero	NF == 0
0100	LT	Less than zero	NF == 1
0101	LEQ	Less than or equal to zero	ZF == 1 OR NF == 1
1010	TF	Test flag set	TF == 1
1011	NTF	Test flag not set	TF == 0
1100	LU	Latched underflow	LUF == 1
1101	LV	Latched overflow	LVF == 1
1110	UNC	Unconditional	None
1111	UNCF <sup>(2)</sup>	Unconditional with flag modification	None

<sup>(1)</sup> Values not shown are reserved.

<sup>(2)</sup> This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

No flags affected

**Pipeline**

This is a single-cycle instruction.

**Example**

```

; X is an array of 32-bit floating-point values
; and has len elements. Find the maximum value in
; the array and store it in Result
;
; Note: MCMFP32 and MSWAPF can be replaced by MMAXF32
;
_ClalTask1:
    MMOVI16    MAR1, #_X          ; Start address
    MUI16TOF32 MR0, @_len        ; Length of the array
    MNOF32     ; delay for MAR1 load
    MNOF32     ; delay for MAR1 load
    MMOV32     MR1, *MAR1[2]++   ; MR1 = X0
LOOP
    MMOV32     MR2, *MAR1[2]++   ; MR2 = next element
    MCMFP32    MR2, MR1          ; Compare MR2 with MR1
    MSWAPF     MR1, MR2, GT      ; MR1 = MAX(MR1, MR2)
    MADD32     MR0, MR0, #-1.0   ; Decrement the counter
    MCMFP32    MR0 #0.0         ; Set/clear flags for MBCNDD
    MNOF32
    MNOF32
    MNOF32
    MBCNDD     LOOP, NEQ        ; Branch if not equal to zero
    MMOV32     @_Result, MR1     ; Always executed
    MNOF32
    MNOF32
    MNOF32
    MSTOP
; End of task

```

**See also**

**MTESTTF CNDF**      **Test MSTF Register Flag Condition**
**Operands**


---

CNDF                      condition to test based on MSTF flags

---

**Opcode**

LSW: 0000 0000 0000 cndf  
MSW: 0111 1111 0100 0000

**Description**

Test the CLA floating-point condition and if true, set the MSTF[TF] flag. If the condition is false, clear the MSTF[TF] flag. This is useful for temporarily storing a condition for later use.

```
if (CNDF == true) TF = 1;
else TF = 0;
```

CNDF is one of the following conditions:

Encode <sup>(3)</sup>	CNDF	Description	MSTF Flags Tested
0000	NEQ	Not equal to zero	ZF == 0
0001	EQ	Equal to zero	ZF == 1
0010	GT	Greater than zero	ZF == 0 AND NF == 0
0011	GEQ	Greater than or equal to zero	NF == 0
0100	LT	Less than zero	NF == 1
0101	LEQ	Less than or equal to zero	ZF == 1 OR NF == 1
1010	TF	Test flag set	TF == 1
1011	NTF	Test flag not set	TF == 0
1100	LU	Latched underflow	LUF == 1
1101	LV	Latched overflow	LVF == 1
1110	UNC	Unconditional	None
1111	UNCF <sup>(4)</sup>	Unconditional with flag modification	None

<sup>(3)</sup> Values not shown are reserved.

<sup>(4)</sup> This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	Yes	No	No	No	No

```
TF = 0;
if (CNDF == true) TF = 1;
```

Note: If (CNDF == UNC or UNCF), the TF flag will be set to 1.

**Pipeline**

This is a single-cycle instruction.

**Example**

```

; if (State == 0.1)
;   RampState = RampState || RAMPMASK
; else if (State == 0.01)
;   CoastState = CoastState || COASTMASK
; else
;   SteadyState = SteadyState || STEADYMASK
;
_ClalTask2:
MMOV32   MR0, @_State
MCMPF32  MR0, #0.1      ; Affects flags for 1st MBCNDD (A)
MCMPF32  MR0, #0.01    ; Check used by 2nd MBCNDD (B)
MTESTTF  EQ            ; Store EQ flag in TF for 2nd MBCNDD (B)
MNOF
MBCNDD   _Skip1, NEQ    ; (A) If State != 0.1, go to Skip1
MMOV32   MR1, @_RampState ; Always executed
MMOVXI   MR2, #RAMPMASK ; Always executed
MOR32    MR1, MR2      ; Always executed
MMOV32   @_RampState, MR1 ; Execute if (A) branch not taken
MSTOP    ; end of task if (A) branch not taken

_Skip1:
MMOV32   MR3, @_SteadyState
MMOVXI   MR2, #STEADYMASK
MOR32    MR3, MR2
MBCNDD   _Skip2, NTF    ; (B) if State != .01, go to Skip2
MMOV32   MR1, @_CoastState ; Always executed
MMOVXI   MR2, #COASTMASK ; Always executed
MOR32    MR1, MR2      ; Always executed
MMOV32   @_CoastState, MR1 ; Execute if (B) branch not taken
MSTOP    ; end of task if (B) branch not taken

_Skip2:
MMOV32   @_SteadyState, MR3 ; Executed if (B) branch taken
MSTOP

```

**See also**

---

**MUI16TOF32 MRa, mem16 *Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value***


---

**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
mem16	16-bit source memory location

**Opcode**

```
LSW: mmmmm mmmmm mmmmm mmmmm
MSW: 0111 0101 01aa addr
```

**Description**

When converting F32 to I16/UI16 data format, the MF32TOI16/UI16 operation truncates to zero while the MF32TOI16R/UI16R operation will round to nearest (even) value.

```
MRa = UI16TOF32[mem16];
```

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**
**See also**

[MF32TOI16 MRa, MRb](#)  
[MF32TOI16R MRa, MRb](#)  
[MF32TOUI16 MRa, MRb](#)  
[MF32TOUI16R MRa, MRb](#)  
[MI16TOF32 MRa, MRb](#)  
[MI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, MRb](#)



## MUI16TOF32 MRa, MRb *Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value*

### Operands

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

### Opcode

LSW: 0000 0000 0000 bbaa  
MSW: 0111 1110 1110 0000

### Description

Convert an unsigned 16-bit integer to a 32-bit floating-point value. When converting float32 to I16/UI16 data format, the MF32TOI16/UI16 operation truncates to zero while the MF32TOI16R/UI16R operation will round to nearest (even) value.

MRa = UI16TOF32[MRb];

### Flags

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

### Pipeline

This is a single-cycle instruction.

### Example

```
MMOVXI MR1, #0x800F ; MR1(15:0) = 32783 (0x800F)
MUI16TOF32 MR0, MR1 ; MR0 = UI16TOF32 (MR1(15:0))
                    ; = 32783.0 (0x47000F00)
```

### See also

[MF32TOI16 MRa, MRb](#)  
[MF32TOI16R MRa, MRb](#)  
[MF32TOUI16 MRa, MRb](#)  
[MF32TOUI16R MRa, MRb](#)  
[MI16TOF32 MRa, MRb](#)  
[MI16TOF32 MRa, mem16](#)  
[MUI16TOF32 MRa, mem16](#)

**MUI32TOF32 MRa, mem32 *Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value***
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
mem32	32-bit memory location accessed using one of the available addressing modes

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0100 10aa addr
```

**Description**

```
MRa = UI32TOF32[mem32];
```

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given x2, m2 and b2 are Uint32 numbers:
;
; x2 = Uint32(2) = 0x00000002
; m2 = Uint32(1) = 0x00000001
; b2 = Uint32(3) = 0x00000003
;
; Calculate y2 = x2 * m2 + b2
;
_ClalTask1:
    MUI32TOF32 MR0, @_m2      ; MR0 = 1.0 (0x3F800000)
    MUI32TOF32 MR1, @_x2      ; MR1 = 2.0 (0x40000000)
    MUI32TOF32 MR2, @_b2      ; MR2 = 3.0 (0x40400000)
    MMPYF32 MR3, MR0, MR1     ; M*X
    MADD32 MR3, MR2, MR3      ; Y=MX+B = 5.0 (0x40A00000)
    MF32TOUI32 MR3, MR3       ; Y = Uint32(5.0) = 0x00000005
    MMOV32 @_y2, MR3          ; store result
    MSTOP                      ; end of task
```

**See also**

[MF32TOI32 MRa, MRb](#)  
[MF32TOUI32 MRa, MRb](#)  
[MI32TOF32 MRa, mem32](#)  
[MI32TOF32 MRa, MRb](#)  
[MUI32TOF32 MRa, MRb](#)

**MUI32TOF32 MRa, MRb *Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value***
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1100 0000
```

**Description**

```
MRa = UI32TOF32 [MRb];
```

**Flags**

This instruction does not affect any flags:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	No	No	No	No

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ    MR3, #0x8000 ; MR3(31:16) = 0x8000
MMOVXI    MR3, #0x1111 ; MR3(15:0) = 0x1111
           ; MR3 = 2147488017
MUI32TOF32 MR3, MR3    ; MR3 = MUI32TOF32 (MR3) = 2147488017.0 (0x4F000011)
```

**See also**

[MF32TOI32 MRa, MRb](#)  
[MF32TOUI32 MRa, MRb](#)  
[MI32TOF32 MRa, mem32](#)  
[MI32TOF32 MRa, MRb](#)  
[MUI32TOF32 MRa, mem32](#)

**MXOR32 MRa, MRb, MRc** *Bitwise Exclusive Or*
**Operands**

MRa	CLA floating-point destination register (MR0 to MR3)
MRb	CLA floating-point source register (MR0 to MR3)
MRc	CLA floating-point source register (MR0 to MR3)

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 1010 0000
```

**Description**

Bitwise XOR of MRb with MRc.

```
MARa(31:0) = MARb(31:0) XOR MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

Flag	TF	ZF	NF	LUF	LVF
Modified	No	Yes	Yes	No	No

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ MR0, #0x5555 ; MR0 = 0x5555AAAA
MMOVXI MR0, #0xAAAA

MMOVIZ MR1, #0x5432 ; MR1 = 0x5432FEDC
MMOVXI MR1, #0xFEDC

; 0101 XOR 0101 = 0000 (0)
; 0101 XOR 0100 = 0001 (1)
; 0101 XOR 0011 = 0110 (6)
; 0101 XOR 0010 = 0111 (7)
; 1010 XOR 1111 = 0101 (5)
; 1010 XOR 1110 = 0100 (4)
; 1010 XOR 1101 = 0111 (7)
; 1010 XOR 1100 = 0110 (6)

MXOR32 MR2, MR1, MR0 ; MR3 = 0x01675476
```

**See also**

[MAND32 MRa, MRb, MRc](#)  
[MOR32 MRa, MRb, MRc](#)

## 5.8 CLA Registers

This section describes the Control Law Accelerator registers.

### 5.8.1 CLA Base Addresses

**Table 5-23. CLA Base Address Table**

Device Registers	Register Name	Start Address	End Address
Cla1OnlyRegs	CLA_ONLY_REGS	0x0000_0C00	0x0000_0CFF
Cla1SoftIntRegs	CLA_SOFTINT_REGS	0x0000_0CE0	0x0000_0CFF
Cla1Regs	CLA_REGS	0x0000_1400	0x0000_147F

## 5.8.2 CLA\_ONLY\_REGS Registers

Table 5-24 lists the CLA\_ONLY\_REGS registers. All register offset addresses not listed in Table 5-24 should be considered as reserved locations and the register contents should not be modified.

**Table 5-24. CLA\_ONLY\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
80h	_MVECTBGRNDACTIVE	Active register for MVECTBGRND.	EALLOW	<a href="#">Go</a>
C0h	_MPSACTL	CLA PSA Control Register	EALLOW	<a href="#">Go</a>
C2h	_MPSA1	CLA PSA1 Register	EALLOW	<a href="#">Go</a>
C4h	_MPSA2	CLA PSA2 Register	EALLOW	<a href="#">Go</a>
E0h	SOFTINTEN	CLA Software Interrupt Enable Register		<a href="#">Go</a>
E2h	SOFTINTFRC	CLA Software Interrupt Force Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 5-25 shows the codes that are used for access types in this section.

**Table 5-25. CLA\_ONLY\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R -0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W 1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 5.8.2.1 \_MVECTBGRNDACTIVE Register (Offset = 80h) [reset = 0h]

\_MVECTBGRNDACTIVE is shown in [Figure 5-2](#) and described in [Table 5-26](#).

Return to the [Summary Table](#).

Gives the current interrupted MPC value of the background task, if the background task was running and interrupted, or reflects the MVECTBGRND value, if MCTLBGRND.BGSTART bit is 0.

**Figure 5-2. \_MVECTBGRNDACTIVE Register**

15	14	13	12	11	10	9	8
i16							
R-0h							
7	6	5	4	3	2	1	0
i16							
R-0h							

**Table 5-26. \_MVECTBGRNDACTIVE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	i16	R	0h	Gives the current interrupted MPC value of the background task, if the background task was running and interrupted, or reflects the MVECTBGRND value, if MCTLBGRND.BGSTART bit is 0. Reset type: SYSRSn

**5.8.2.2 \_MPSACTL Register (Offset = C0h) [reset = 0h]**

\_MPSACTL is shown in [Figure 5-3](#) and described in [Table 5-27](#).

Return to the [Summary Table](#).

PSA Control Register

**Figure 5-3. \_MPSACTL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
MPSA2CFG		MPSA2CLEAR	MPSA1CLEAR	MDWDBCYC	MDWDBSTAR T	MPABCYC	MPABSTART
R/W-0h		R-0/W1S-0h	R-0/W1S-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 5-27. \_MPSACTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-6	MPSA2CFG	R/W	0h	CLA PSA2 Polynomial Configuration Bits: These bits configure the type of polynomial used for PSA2. The polynomials chosen are commonly used in the industry: Mode Polynomial Type 0,0 PSA 0,1 CRC32 1,0 CRC16 1,1 CRC16-CCITT  Note: [1] Polynomial configuration should be performed when PSA2 is stopped. Reset type: SYSRSn
5	MPSA2CLEAR	R-0/W1S	0h	CLA PSA2 Clear Bit: Writing of "1" will clear contents of PSA2 register. Writes of "0" are ignored. Always reads back a "0"  Note: Clearing operation should be performed when PSA2 is stopped. Reset type: SYSRSn
4	MPSA1CLEAR	R-0/W1S	0h	CLA PSA1 Clear Bit: Writing of "1" will clear contents of PSA1 register. Writes of "0" are ignored. Always reads back a "0"  Note: Clearing operation should be performed when PSA1 is stopped. Reset type: SYSRSn
3	MDWDBCYC	R/W	0h	CLA Data Write Data Bus PSA2 Cycle or Event Based Bit: 0 PSA2 calculated on every cycle 1 PSA2 calculated on every bus event Reset type: SYSRSn
2	MDWDBSTART	R/W	0h	CLA Data Write Data Bus PSA2 Start/Stop Bit: 0 PSA2 stopped 1 PSA2 start Reset type: SYSRSn



**Table 5-27. \_MPSACTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	MPABCYC	R/W	0h	CLA Program Address Bus PSA1 Cycle/Event Based Bit: 0 PSA1 calculated on every cycle 1 PSA1 calculated on every bus event Reset type: SYSRSn
0	MPABSTART	R/W	0h	CLA Program Address Bus PSA1 Start/Stop Bit: 0 PSA1 stopped 1 PSA1 start Reset type: SYSRSn

### 5.8.2.3 \_MPSA1 Register (Offset = C2h) [reset = 0h]

\_MPSA1 is shown in [Figure 5-4](#) and described in [Table 5-28](#).

Return to the [Summary Table](#).

PSA1 Register

**Figure 5-4. \_MPSA1 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
i32																															
R/W-0h																															

**Table 5-28. \_MPSA1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	i32	R/W	0h	PSA1 Value: Reading this register gives the current PSA1 value. The value can be read at any time.  Writes to this register are allowed to initialize the PSA1 to a known value. Writes to this register should only be made when PSA1 is stopped.  Register value is cleared to zero by reset or by writing to the MPSA1CLEAR bit in the MPSACTL register. Reset type: SYSRSn

#### 5.8.2.4 \_MPSA2 Register (Offset = C4h) [reset = 0h]

\_MPSA2 is shown in [Figure 5-5](#) and described in [Table 5-29](#).

Return to the [Summary Table](#).

PSA2 Register

**Figure 5-5. \_MPSA2 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
i32																															
R/W-0h																															

**Table 5-29. \_MPSA2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	i32	R/W	0h	<p>PSA2 Value: Reading this register gives the current PSA2 value. The value can be read at any time.</p> <p>Writes to this register are allowed to initialize the PSA2 to a known value. Writes to this register should only be made when PSA2 is stopped.</p> <p>Register value is cleared to zero by reset or by writing to the MPSA2CLEAR bit in the MPSACTL register.</p> <p>Reset type: SYSRSn</p>

### 5.8.2.5 SOFTINTEN Register (Offset = E0h) [reset = 0h]

SOFTINTEN is shown in [Figure 5-6](#) and described in [Table 5-30](#).

Return to the [Summary Table](#).

Enables the ability to generate CLA task interrupt from within the task, by writing to SOFTINTFRC register. SOFTINTFRC register can only be written from CLA.

**Figure 5-6. SOFTINTEN Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
TASK8	TASK7	TASK6	TASK5	TASK4	TASK3	TASK2	TASK1
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 5-30. SOFTINTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	TASK8	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
6	TASK7	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
5	TASK6	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
4	TASK5	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
3	TASK4	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
2	TASK3	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
1	TASK2	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn

**Table 5-30. SOFTINTEN Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	TASK1	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case.  Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn

### 5.8.2.6 SOFTINTFRC Register (Offset = E2h) [reset = 0h]

SOFTINTFRC is shown in [Figure 5-7](#) and described in [Table 5-31](#).

Return to the [Summary Table](#).

Writing a value of 1 in a bit will generate the corresponding task interrupt.

**Figure 5-7. SOFTINTFRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
TASK8	TASK7	TASK6	TASK5	TASK4	TASK3	TASK2	TASK1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 5-31. SOFTINTFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	TASK8	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
6	TASK7	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
5	TASK6	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
4	TASK5	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
3	TASK4	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
2	TASK3	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
1	TASK2	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
0	TASK1	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn

### 5.8.3 CLA\_SOFTINT\_REGS Registers

Table 5-32 lists the CLA\_SOFTINT\_REGS registers. All register offset addresses not listed in Table 5-32 should be considered as reserved locations and the register contents should not be modified.

**Table 5-32. CLA\_SOFTINT\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	SOFTINTEN	CLA Software Interrupt Enable Register		<a href="#">Go</a>
2h	SOFTINTFRC	CLA Software Interrupt Force Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 5-33 shows the codes that are used for access types in this section.

**Table 5-33. CLA\_SOFTINT\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 5.8.3.1 SOFTINTEN Register (Offset = 0h) [reset = 0h]

SOFTINTEN is shown in [Figure 5-8](#) and described in [Table 5-34](#).

Return to the [Summary Table](#).

Enables the ability to generate CLA task interrupt from within the task, by writing to SOFTINTFRC register. SOFTINTFRC register can only be written from CLA.

**Figure 5-8. SOFTINTEN Register**

15								14								13								12								11								10								9								8							
RESERVED																																																															
R/W-0h																																																															
7								6								5								4								3								2								1								0							
TASK8								TASK7								TASK6								TASK5								TASK4								TASK3								TASK2								TASK1							
R/W-0h								R/W-0h								R/W-0h								R/W-0h								R/W-0h								R/W-0h								R/W-0h															

**Table 5-34. SOFTINTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	TASK8	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
6	TASK7	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
5	TASK6	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
4	TASK5	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
3	TASK4	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
2	TASK3	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
1	TASK2	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn



**Table 5-34. SOFTINTEN Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	TASK1	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn

### 5.8.3.2 SOFTINTFRC Register (Offset = 2h) [reset = 0h]

SOFTINTFRC is shown in [Figure 5-9](#) and described in [Table 5-35](#).

Return to the [Summary Table](#).

Writing a value of 1 in a bit will generate the corresponding task interrupt. This register is only accessible by the CLA (not the CPU).

**Figure 5-9. SOFTINTFRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
TASK8	TASK7	TASK6	TASK5	TASK4	TASK3	TASK2	TASK1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 5-35. SOFTINTFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	TASK8	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
6	TASK7	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
5	TASK6	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
4	TASK5	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
3	TASK4	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
2	TASK3	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
1	TASK2	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn
0	TASK1	R-0/W1S	0h	Write of '1' will generate a CLA software interrupt to the CPU for the corresponding task.  Write of '0' has no effect. Reset type: SYSRSn

### 5.8.4 CLA\_REGS Registers

Table 5-36 lists the CLA\_REGS registers. All register offset addresses not listed in Table 5-36 should be considered as reserved locations and the register contents should not be modified.

**Table 5-36. CLA\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	MVECT1	Task Interrupt Vector	EALLOW	<a href="#">Go</a>
1h	MVECT2	Task Interrupt Vector	EALLOW	<a href="#">Go</a>
2h	MVECT3	Task Interrupt Vector	EALLOW	<a href="#">Go</a>
3h	MVECT4	Task Interrupt Vector	EALLOW	<a href="#">Go</a>
4h	MVECT5	Task Interrupt Vector	EALLOW	<a href="#">Go</a>
5h	MVECT6	Task Interrupt Vector	EALLOW	<a href="#">Go</a>
6h	MVECT7	Task Interrupt Vector	EALLOW	<a href="#">Go</a>
7h	MVECT8	Task Interrupt Vector	EALLOW	<a href="#">Go</a>
10h	MCTL	Control Register	EALLOW	<a href="#">Go</a>
1Bh	_MVECTBGRNDACTIVE	Active register for MVECTBGRND.	EALLOW	<a href="#">Go</a>
1Ch	SOFTINTEN	CLA Software Interrupt Enable Register		<a href="#">Go</a>
1Dh	_MSTSBGRND	Status register for the back ground task.	EALLOW	<a href="#">Go</a>
1Eh	_MCTLBGRND	Control register for the back ground task.	EALLOW	<a href="#">Go</a>
1Fh	_MVECTBGRND	Vector for the back ground task.	EALLOW	<a href="#">Go</a>
20h	MIFR	Interrupt Flag Register	EALLOW	<a href="#">Go</a>
21h	MIOVF	Interrupt Overflow Flag Register	EALLOW	<a href="#">Go</a>
22h	MIFRC	Interrupt Force Register	EALLOW	<a href="#">Go</a>
23h	MICLR	Interrupt Flag Clear Register	EALLOW	<a href="#">Go</a>
24h	MICLROVF	Interrupt Overflow Flag Clear Register	EALLOW	<a href="#">Go</a>
25h	MIER	Interrupt Enable Register	EALLOW	<a href="#">Go</a>
26h	MIRUN	Interrupt Run Status Register	EALLOW	<a href="#">Go</a>
28h	_MPC	CLA Program Counter		<a href="#">Go</a>
2Ah	_MAR0	CLA Auxiliary Register 0		<a href="#">Go</a>
2Bh	_MAR1	CLA Auxiliary Register 1		<a href="#">Go</a>
2Eh	_MSTF	CLA Floating-Point Status Register		<a href="#">Go</a>
30h	_MR0	CLA Floating-Point Result Register 0		<a href="#">Go</a>
34h	_MR1	CLA Floating-Point Result Register 1		<a href="#">Go</a>
38h	_MR2	CLA Floating-Point Result Register 2		<a href="#">Go</a>
3Ch	_MR3	CLA Floating-Point Result Register 3		<a href="#">Go</a>
42h	_MPSACTL	CLA PSA Control Register	EALLOW	<a href="#">Go</a>
44h	_MPSA1	CLA PSA1 Register	EALLOW	<a href="#">Go</a>
46h	_MPSA2	CLA PSA2 Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 5-37 shows the codes that are used for access types in this section.

**Table 5-37. CLA\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write

**Table 5-37. CLA\_REGS Access Type Codes (continued)**

Access Type	Code	Description
W1C	W 1C	Write 1 to clear
W1S	W 1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 5.8.4.1 MVECT1 Register (Offset = 0h) [reset = 0h]

MVECT1 is shown in [Figure 5-10](#) and described in [Table 5-38](#).

Return to the [Summary Table](#).

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 5-10. MVECT1 Register**

15	14	13	12	11	10	9	8
MVECT							
R/W-0h							
7	6	5	4	3	2	1	0
MVECT							
R/W-0h							

**Table 5-38. MVECT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	MVECT	R/W	0h	<p>MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.</p> <p>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.</p> <p>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..</p> <p>Reset type: SYSRSn</p>

### 5.8.4.2 MVECT2 Register (Offset = 1h) [reset = 0h]

MVECT2 is shown in [Figure 5-11](#) and described in [Table 5-39](#).

Return to the [Summary Table](#).

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 5-11. MVECT2 Register**

15	14	13	12	11	10	9	8
MVECT							
R/W-0h							
7	6	5	4	3	2	1	0
MVECT							
R/W-0h							

**Table 5-39. MVECT2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	MVECT	R/W	0h	<p>MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.</p> <p>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.</p> <p>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..</p> <p>Reset type: SYRSn</p>

### 5.8.4.3 MVECT3 Register (Offset = 2h) [reset = 0h]

MVECT3 is shown in [Figure 5-12](#) and described in [Table 5-40](#).

Return to the [Summary Table](#).

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 5-12. MVECT3 Register**

15	14	13	12	11	10	9	8
MVECT							
R/W-0h							
7	6	5	4	3	2	1	0
MVECT							
R/W-0h							

**Table 5-40. MVECT3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	MVECT	R/W	0h	<p>MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.</p> <p>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.</p> <p>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..</p> <p>Reset type: SYSRSn</p>

#### 5.8.4.4 MVECT4 Register (Offset = 3h) [reset = 0h]

MVECT4 is shown in [Figure 5-13](#) and described in [Table 5-41](#).

Return to the [Summary Table](#).

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 5-13. MVECT4 Register**

15	14	13	12	11	10	9	8
MVECT							
R/W-0h							
7	6	5	4	3	2	1	0
MVECT							
R/W-0h							

**Table 5-41. MVECT4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	MVECT	R/W	0h	<p>MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.</p> <p>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.</p> <p>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..</p> <p>Reset type: SYRSn</p>



### 5.8.4.5 MVECT5 Register (Offset = 4h) [reset = 0h]

MVECT5 is shown in [Figure 5-14](#) and described in [Table 5-42](#).

Return to the [Summary Table](#).

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 5-14. MVECT5 Register**

15	14	13	12	11	10	9	8
MVECT							
R/W-0h							
7	6	5	4	3	2	1	0
MVECT							
R/W-0h							

**Table 5-42. MVECT5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	MVECT	R/W	0h	<p>MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.</p> <p>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.</p> <p>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..</p> <p>Reset type: SYSRSn</p>

#### 5.8.4.6 MVECT6 Register (Offset = 5h) [reset = 0h]

MVECT6 is shown in [Figure 5-15](#) and described in [Table 5-43](#).

Return to the [Summary Table](#).

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 5-15. MVECT6 Register**

15	14	13	12	11	10	9	8
MVECT							
R/W-0h							
7	6	5	4	3	2	1	0
MVECT							
R/W-0h							

**Table 5-43. MVECT6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	MVECT	R/W	0h	<p>MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.</p> <p>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.</p> <p>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..</p> <p>Reset type: SYRSn</p>

### 5.8.4.7 MVECT7 Register (Offset = 6h) [reset = 0h]

MVECT7 is shown in [Figure 5-16](#) and described in [Table 5-44](#).

Return to the [Summary Table](#).

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 5-16. MVECT7 Register**

15	14	13	12	11	10	9	8
MVECT							
R/W-0h							
7	6	5	4	3	2	1	0
MVECT							
R/W-0h							

**Table 5-44. MVECT7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	MVECT	R/W	0h	<p>MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.</p> <p>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.</p> <p>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..</p> <p>Reset type: SYSRSn</p>

#### 5.8.4.8 MVECT8 Register (Offset = 7h) [reset = 0h]

MVECT8 is shown in [Figure 5-17](#) and described in [Table 5-45](#).

Return to the [Summary Table](#).

Each CLA interrupt has its own interrupt vector (MVECT1 to MVECT8). This interrupt vector points to the first instruction of the associated task. When a task begins, the CLA will start fetching instructions at the location indicated by the appropriate MVECT register .

**Figure 5-17. MVECT8 Register**

15	14	13	12	11	10	9	8
MVECT							
R/W-0h							
7	6	5	4	3	2	1	0
MVECT							
R/W-0h							

**Table 5-45. MVECT8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	MVECT	R/W	0h	<p>MPC Start Address: These bits specify the start address for the given interrupt (task). The address range of the CLA with a 16-bit MVECT is 64Kx16 words or 32K CLA instructions.</p> <p>There is one MVECT register per interrupt (task). Interrupt 1 uses MVECT1, interrupt 2 uses MVECT2 and so forth.</p> <p>Note: While the CLA is running or executing a task, the CPU can change the MVECT values..</p> <p>Reset type: SYRSn</p>

### 5.8.4.9 MCTL Register (Offset = 10h) [reset = 0h]

MCTL is shown in [Figure 5-18](#) and described in [Table 5-46](#).

Return to the [Summary Table](#).

Control Register

**Figure 5-18. MCTL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					IACKE	SOFTRESET	HARDRESET
R-0h					R/W-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 5-46. MCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	IACKE	R/W	0h	<p>IACK Operation Enable Bit: Writing a "1" to this bit will enable the IACK operation for setting the MIFR bits in the same manner as the MIFRC register (write of "1" will set respective MIFR bit). At reset, this feature is disabled.</p> <p>This feature enables the C28 CPU to efficiently trigger a task.</p> <p>Note: IACK operation should ignore EALLOW status of C28 core when accessing the MIFRC register.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The CLA ignores the IACK instruction. (default)</p> <p>1h (R/W) = Enable the main CPU to use the IACK #16bit instruction to set MIFR bits in the same manner as writing to the MIFRC register. Each bit in the operand, #16bit, corresponds to a bit in the MIFRC register. Using IACK has the advantage of not having to first set the EALLOW bit. This allows the main CPU to efficiently trigger a CLA task through software.</p> <p>Examples IACK #0x0001 Write a 1 to MIFRC bit 0 to force task 1 IACK #0x0003 Write a 1 to MIFRC bit 0 and 1 to force task 1 and task 2</p>
1	SOFTRESET	R-0/W1S	0h	<p>Soft Reset Bit: Writing a "1" to this bit will stop a current task, clear the RUN flag and also clear all bits in the MIER register. Writes of "0" are ignored and reads always return a "0".</p> <p>Note: After issuing SOFTRESET command, user should wait at least 1 clock cycle before attempting to write to MIER register.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 are ignored.</p> <p>1h (R/W) = Writing a 1 will cause a soft reset of the CLA. This will stop the current task, clear the MIRUN flag and clear all bits in the MIER register. After a soft reset you must wait at least 1 SYSCLKOUT cycle before reconfiguring the MIER bits. If these two operations are done back-to-back then the MIER bits will not get set.</p>
0	HARDRESET	R-0/W1S	0h	<p>Hard Reset Bit: Writing a "1" to this bit will cause a HARD reset on the CLA. The behavior of a HARD reset is the same as a system reset SYSRSn on the CLA. Writes of "0" are ignored and reads always return a "0".</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 are ignored.</p> <p>1h (R/W) = Writing a 1 will cause a hard reset of the CLA. This will set all CLA registers to their default state.</p>

#### 5.8.4.10 \_MVECTBGRNDACTIVE Register (Offset = 1Bh) [reset = 0h]

\_MVECTBGRNDACTIVE is shown in [Figure 5-19](#) and described in [Table 5-47](#).

Return to the [Summary Table](#).

Gives the current interrupted MPC value of the background task, if the background task was running and interrupted, or reflects the MVECTBGRND value, if MCTLBGRND.BGSTART bit is 0.

**Figure 5-19. \_MVECTBGRNDACTIVE Register**

15	14	13	12	11	10	9	8
i16							
R-0h							
7	6	5	4	3	2	1	0
i16							
R-0h							

**Table 5-47. \_MVECTBGRNDACTIVE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	i16	R	0h	Gives the current interrupted MPC value of the background task, if the background task was running and interrupted, or reflects the MVECTBGRND value, if MCTLBGRND.BGSTART bit is 0. Reset type: SYSRSn

### 5.8.4.11 SOFTINTEN Register (Offset = 1Ch) [reset = 0h]

SOFTINTEN is shown in [Figure 5-20](#) and described in [Table 5-48](#).

Return to the [Summary Table](#).

Enables the ability to generate CLA task interrupt from within the task, by writing to SOFTINTFRC register. SOFTINTFRC register can only be written from CLA. Only reads are allowed from CPU. Writes are not allowed from CPU.

**Figure 5-20. SOFTINTEN Register**

15								14								13								12								11								10								9								8							
RESERVED																																																															
R/W-0h																																																															
7								6								5								4								3								2								1								0							
TASK8								TASK7								TASK6								TASK5								TASK4								TASK3								TASK2								TASK1							
R/W-0h								R/W-0h								R/W-0h								R/W-0h								R/W-0h								R/W-0h								R/W-0h															

**Table 5-48. SOFTINTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	TASK8	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
6	TASK7	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
5	TASK6	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
4	TASK5	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
3	TASK4	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
2	TASK3	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn

**Table 5-48. SOFTINTEN Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	TASK2	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn
0	TASK1	R/W	0h	0: End-Of-Task Interrupt is fired for the respective task 1: Enable Software Interrupt for the respective task. End-of-Task interrupt is not sent to CPU in this case. Note: SOFTINTEN register is read only in the CPU memory map. Reset type: SYSRSn



**5.8.4.12 \_MSTSBGRND Register (Offset = 1Dh) [reset = 0h]**

\_MSTSBGRND is shown in [Figure 5-21](#) and described in [Table 5-49](#).

Return to the [Summary Table](#).

Status bits for the backgroundtask.

**Figure 5-21. \_MSTSBGRND Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED					BGOVF	_BGINTM	RUN
R/W-0h					R/W1C-0h	R-0h	R-0h

**Table 5-49. \_MSTSBGRND Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R/W	0h	Reserved
2	BGOVF	R/W1C	0h	Value of 1 indicates a hardware trigger (which is enabled) occurred while the MCTLBGRND.BGSTART bit is set. Writing a value of 1 to this bit clears the BGOVF bit. Write of 0 has no effect, Value of 0 indicates the background task trigger did not result in a overflow. Reset type: SYSRSn
1	_BGINTM	R	0h	Value of 1 indicates that background task will not be interrupted. This bit is set when MSETC _BGINTM bit is executed. Value of 0 indicates that background task can be interrupted. Reset type: SYSRSn
0	RUN	R	0h	Value of 1 indicates that background task is running. Value of 0 indicates that background task is not running. Reset type: SYSRSn

**5.8.4.13 \_MCTLBGRND Register (Offset = 1Eh) [reset = 0h]**

\_MCTLBGRND is shown in [Figure 5-22](#) and described in [Table 5-50](#).

Return to the [Summary Table](#).

Holds the configuration bits to start the background task, enable hardware trigger.

**Figure 5-22. \_MCTLBGRND Register**

15	14	13	12	11	10	9	8
BGEN		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
RESERVED						TRIGEN	BGSTART
R/W-0h						R/W-0h	R/W1S-0h

**Table 5-50. \_MCTLBGRND Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	BGEN	R/W	0h	0 Background task is disabled, BGSTART will not be set either in a hardware trigger or by writing 1 to BGSTART bit. 1 Background task is enabled and MIER[INT8] will be cleared, preventing task 8 from triggering. Reset type: SYSRSn
14-2	RESERVED	R/W	0h	Reserved
1	TRIGEN	R/W	0h	Hardware trigger enable for the background task. 1 Hardware trigger is enabled. 0 Hardware trigger is disabled. Note: Trigger source for the background task will be the same as that for task 8 Reset type: SYSRSn
0	BGSTART	R/W1S	0h	Value of 1 will start the background task, provided there are no other pending tasks. - Value of 0 has no effect if the background task has not started. - This bit is also set by hardware, if MCTLBGRND.TRIGEN = 1 and a hardware trigger occurs. - This bit is cleared by hardware when a MSTOP instruction occurs in the background task - If the background task is running and this bit is cleared, it will not have any effect on the task execution. Reset type: SYSRSn

**5.8.4.14 \_MVECTBGRND Register (Offset = 1Fh) [reset = 0h]**

\_MVECTBGRND is shown in [Figure 5-23](#) and described in [Table 5-51](#).

Return to the [Summary Table](#).

These bits specify the start address for the background task . The value in this register is forced into the MPC register when the background task starts.

**Figure 5-23. \_MVECTBGRND Register**

15	14	13	12	11	10	9	8
i16							
R/W-0h							
7	6	5	4	3	2	1	0
i16							
R/W-0h							

**Table 5-51. \_MVECTBGRND Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	i16	R/W	0h	MPC Start Address: These bits specify the start address for the background task . The value in this register is forced into the MPC register, when the background task starts. Reset type: SYSRSn

### 5.8.4.15 MIFR Register (Offset = 20h) [reset = 0h]

MIFR is shown in [Figure 5-24](#) and described in [Table 5-52](#).

Return to the [Summary Table](#).

Each bit in the interrupt flag register corresponds to a CLA task. The corresponding bit is automatically set when the task request is received from the peripheral interrupt. The bit can also be set by the main CPU writing to the MIFRC register or using the IACK instruction to start the task. To use the IACK instruction to begin a task first enable this feature in the MCTL register. If the bit is already set when a new peripheral interrupt is received, then the corresponding overflow bit will be set in the MIOVF register.

The corresponding MIFR bit is automatically cleared when the task begins execution. This will occur if the interrupt is enabled in the MIER register and no other higher priority task is pending. The bits can also be cleared manually by writing to the MICLR register. Writes to the MIFR register are ignored.

**Figure 5-24. MIFR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 5-52. MIFR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R	0h	<p>These bits, when set to "1", indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.</p> <p>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to "1" while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.</p> <p>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.</p> <p>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.</p> <p>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.</p> <p>Reset type: SYSRSn                      0h (R/W) = TASK_FLAG_DISABLE                      Task 8 interrupt is currently not flagged (default)                      1h (R/W) = TASK_FLAG_ENABLE                      Task 8 interrupt has been received and is pending execution</p>

**Table 5-52. MIFR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	INT7	R	0h	<p>These bits, when set to "1", indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.</p> <p>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to "1" while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.</p> <p>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.</p> <p>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.</p> <p>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.</p> <p>Reset type: SYSRSn            0h (R/W) = TASK_FLAG_DISABLE            Task 7 interrupt is currently not flagged (default)            1h (R/W) = TASK_FLAG_ENABLE            Task 7 interrupt has been received and is pending execution</p>
5	INT6	R	0h	<p>These bits, when set to "1", indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.</p> <p>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to "1" while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.</p> <p>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.</p> <p>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.</p> <p>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.</p> <p>Reset type: SYSRSn            0h (R/W) = TASK_FLAG_DISABLE            Task 6 interrupt is currently not flagged (default)            1h (R/W) = TASK_FLAG_ENABLE            Task 6 interrupt has been received and is pending execution</p>

**Table 5-52. MIFR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	INT5	R	0h	<p>These bits, when set to "1", indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.</p> <p>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to "1" while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.</p> <p>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.</p> <p>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.</p> <p>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.</p> <p>Reset type: SYSRSn                      0h (R/W) = TASK_FLAG_DISABLE                      Task 5 interrupt is currently not flagged (default)                      1h (R/W) = TASK_FLAG_ENABLE                      Task 5 interrupt has been received and is pending execution</p>
3	INT4	R	0h	<p>These bits, when set to "1", indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.</p> <p>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to "1" while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.</p> <p>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.</p> <p>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.</p> <p>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.</p> <p>Reset type: SYSRSn                      0h (R/W) = TASK_FLAG_DISABLE                      Task 4 interrupt is currently not flagged (default)                      1h (R/W) = TASK_FLAG_ENABLE                      Task 4 interrupt has been received and is pending execution</p>

**Table 5-52. MIFR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INT3	R	0h	<p>These bits, when set to "1", indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.</p> <p>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to "1" while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.</p> <p>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.</p> <p>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.</p> <p>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.</p> <p>Reset type: SYSRSn            0h (R/W) = TASK_FLAG_DISABLE            Task 3 interrupt is currently not flagged (default)            1h (R/W) = TASK_FLAG_ENABLE            Task 3 interrupt has been received and is pending execution</p>
1	INT2	R	0h	<p>These bits, when set to "1", indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.</p> <p>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to "1" while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.</p> <p>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.</p> <p>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.</p> <p>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.</p> <p>Reset type: SYSRSn            0h (R/W) = TASK_FLAG_DISABLE            Task 2 interrupt is currently not flagged (default)            1h (R/W) = TASK_FLAG_ENABLE            Task 2 interrupt has been received and is pending execution</p>

**Table 5-52. MIFR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	INT1	R	0h	<p>These bits, when set to "1", indicate a valid peripheral interrupt has been latched by the CLA. Writes to this register are ignored.</p> <p>The IFR flag bit is automatically cleared if the respective interrupt is enabled in the MIER register and the respective task starts running. If a new peripheral interrupt attempts to set the bit to "1" while on the same cycle the task tries to clear it, then the peripheral interrupt will have priority.</p> <p>The IFR flag bits can also be set and cleared by the MIFRC and MICLR registers.</p> <p>If the MIFRC register is trying to set the respective bit while a new task tries to clear it, then the MIFRC event has priority.</p> <p>If the MICLR register is trying to clear the respective bit and a peripheral interrupt occurs on the same cycle, then the peripheral interrupt has priority. The respective overflow flag in the MIOVF register will not be set under this condition.</p> <p>Reset type: SYSRSn                      0h (R/W) = TASK_FLAG_DISABLE                      Task 1 interrupt is currently not flagged (default)                      1h (R/W) = TASK_FLAG_ENABLE                      Task 1 interrupt has been received and is pending execution</p>



### 5.8.4.16 MIOVF Register (Offset = 21h) [reset = 0h]

MIOVF is shown in [Figure 5-25](#) and described in [Table 5-53](#).

Return to the [Summary Table](#).

Each bit in the overflow flag register corresponds to a CLA task. The bit is set when an interrupt overflow event has occurred for the specific task. An overflow event occurs when the MIFR register bit is already set when a new interrupt is received from a peripheral source. The MIOVF bits are only affected by peripheral interrupt events. They do not respond to a task request by the main CPU IACK instruction or by directly setting MIFR bits. The overflow flag will remain latched and can only be cleared by writing to the overflow flag clear (MICLROVF) register. Writes to the MIOVF register are ignored.

**Figure 5-25. MIOVF Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 5-53. MIOVF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R	0h	<p>These bits, when set to "1", indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.</p> <p>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.</p> <p>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.</p> <p>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.</p> <p>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.</p> <p>Reset type: SYSRSn                      0h (R/W) = A task 8 interrupt overflow has not occurred (default)                      1h (R/W) = A task 8 interrupt overflow has occurred</p>
6	INT7	R	0h	<p>These bits, when set to "1", indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.</p> <p>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.</p> <p>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.</p> <p>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.</p> <p>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.</p> <p>Reset type: SYSRSn                      0h (R/W) = A task 7 interrupt overflow has not occurred (default)                      1h (R/W) = A task 7 interrupt overflow has occurred</p>

**Table 5-53. MIOVF Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	INT6	R	0h	<p>These bits, when set to "1", indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.</p> <p>An overflow event remains latched and respective bits can only be cleared by writing to the MIOVF register.</p> <p>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.</p> <p>If the MIOVF bit is being cleared by the MIOVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.</p> <p>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.</p> <p>Reset type: SYSRSn                      0h (R/W) = A task 6 interrupt overflow has not occurred (default)                      1h (R/W) = A task 6 interrupt overflow has occurred</p>
4	INT5	R	0h	<p>These bits, when set to "1", indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.</p> <p>An overflow event remains latched and respective bits can only be cleared by writing to the MIOVF register.</p> <p>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.</p> <p>If the MIOVF bit is being cleared by the MIOVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.</p> <p>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.</p> <p>Reset type: SYSRSn                      0h (R/W) = A task 5 interrupt overflow has not occurred (default)                      1h (R/W) = A task 5 interrupt overflow has occurred</p>
3	INT4	R	0h	<p>These bits, when set to "1", indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.</p> <p>An overflow event remains latched and respective bits can only be cleared by writing to the MIOVF register.</p> <p>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.</p> <p>If the MIOVF bit is being cleared by the MIOVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.</p> <p>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.</p> <p>Reset type: SYSRSn                      0h (R/W) = A task 4 interrupt overflow has not occurred (default)                      1h (R/W) = A task 4 interrupt overflow has occurred</p>

**Table 5-53. MIOVF Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INT3	R	0h	<p>These bits, when set to "1", indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.</p> <p>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.</p> <p>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.</p> <p>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.</p> <p>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.</p> <p>Reset type: SYSRSn            0h (R/W) = A task 3 interrupt overflow has not occurred (default)            1h (R/W) = A task 3 interrupt overflow has occurred</p>
1	INT2	R	0h	<p>These bits, when set to "1", indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.</p> <p>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.</p> <p>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.</p> <p>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.</p> <p>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.</p> <p>Reset type: SYSRSn            0h (R/W) = A task 2 interrupt overflow has not occurred (default)            1h (R/W) = A task 2 interrupt overflow has occurred</p>
0	INT1	R	0h	<p>These bits, when set to "1", indicate an interrupt overflow event occurred. Such an event occurs when the IFR bit is already set.</p> <p>An overflow event remains latched and respective bits can only be cleared by writing to the MICLROVF register.</p> <p>If the MIFR bit is being cleared by a new task on the same cycle as a new peripheral interrupt occurs, the overflow flag will not be affected and the respective MIFR bit will be set.</p> <p>If the MIOVF bit is being cleared by the MICLROVF register on the same cycle as the overflow bit is being set by hardware, then the hardware will have priority.</p> <p>Notes: [1] The MIOVF bits are only affected by peripheral interrupt events. Forcing an interrupt using the MIFRC or IACK operation will not set the overflow flag even if the MIFR bit is set.</p> <p>Reset type: SYSRSn            0h (R/W) = A task 1 interrupt overflow has not occurred (default)            1h (R/W) = A task 1 interrupt overflow has occurred</p>

### 5.8.4.17 MIFRC Register (Offset = 22h) [reset = 0h]

MIFRC is shown in [Figure 5-26](#) and described in [Table 5-54](#).

Return to the [Summary Table](#).

The interrupt force register can be used by the main CPU to start tasks through software. Writing a 1 to a MIFRC bit will set the corresponding bit in the MIFR register. Writes of 0 are ignored and reads always return 0. The IACK #16bit operation can also be used to start tasks and has the same effect as the MIFRC register. To enable IACK to set MIFR bits you must first set the MCTL[IACKE] bit. Using IACK has the advantage of not having to first set the EALLOW bit. This allows the main CPU to efficiently trigger CLA tasks through software.

**Figure 5-26. MIFRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 5-54. MIFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R-0/W1S	0h	Writing a "1" to any of the bits will set the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIFR register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to force the task 8 interrupt
6	INT7	R-0/W1S	0h	Writing a "1" to any of the bits will set the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIFR register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to force the task 7 interrupt
5	INT6	R-0/W1S	0h	Writing a "1" to any of the bits will set the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIFR register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to force the task 6 interrupt
4	INT5	R-0/W1S	0h	Writing a "1" to any of the bits will set the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIFR register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to force the task 5 interrupt

**Table 5-54. MIFRC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	INT4	R-0/W1S	0h	<p>Writing a "1" to any of the bits will set the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIFR register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to force the task 4 interrupt</p>
2	INT3	R-0/W1S	0h	<p>Writing a "1" to any of the bits will set the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIFR register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to force the task 3 interrupt</p>
1	INT2	R-0/W1S	0h	<p>Writing a "1" to any of the bits will set the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIFR register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to force the task 2 interrupt</p>
0	INT1	R-0/W1S	0h	<p>Writing a "1" to any of the bits will set the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIFR register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to force the task 1 interrupt</p>

### 5.8.4.18 MICLR Register (Offset = 23h) [reset = 0h]

MICLR is shown in [Figure 5-27](#) and described in [Table 5-55](#).

Return to the [Summary Table](#).

Normally bits in the MIFR register are automatically cleared when a task begins. The interrupt flag clear register can be used to instead manually clear bits in the interrupt flag (MIFR) register. Writing a 1 to a MICLR bit will clear the corresponding bit in the MIFR register. Writes of 0 are ignored and reads always return 0.

**Figure 5-27. MICLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 5-55. MICLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R-0/W1S	0h	Writing a "1" to any of the bits will clear the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIFR register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to clear the task 8 interrupt flag
6	INT7	R-0/W1S	0h	Writing a "1" to any of the bits will clear the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIFR register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to clear the task 7 interrupt flag
5	INT6	R-0/W1S	0h	Writing a "1" to any of the bits will clear the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIFR register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to clear the task 6 interrupt flag
4	INT5	R-0/W1S	0h	Writing a "1" to any of the bits will clear the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIFR register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to clear the task 5 interrupt flag

**Table 5-55. MICLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	INT4	R-0/W1S	0h	<p>Writing a "1" to any of the bits will clear the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIFR register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to clear the task 4 interrupt flag</p>
2	INT3	R-0/W1S	0h	<p>Writing a "1" to any of the bits will clear the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIFR register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to clear the task 3 interrupt flag</p>
1	INT2	R-0/W1S	0h	<p>Writing a "1" to any of the bits will clear the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIFR register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to clear the task 2 interrupt flag</p>
0	INT1	R-0/W1S	0h	<p>Writing a "1" to any of the bits will clear the corresponding MIFR bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIFR register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to clear the task 1 interrupt flag</p>

### 5.8.4.19 MICLROVF Register (Offset = 24h) [reset = 0h]

MICLROVF is shown in [Figure 5-28](#) and described in [Table 5-56](#).

Return to the [Summary Table](#).

Overflow flag bits in the MIOVF register are latched until manually cleared using the MICLROVF register. Writing a 1 to a MICLROVF bit will clear the corresponding bit in the MIOVF register. Writes of 0 are ignored and reads always return 0.

**Figure 5-28. MICLROVF Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 5-56. MICLROVF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R-0/W1S	0h	Writing a "1" to any of the bits will clear the corresponding MIOVF bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIOVF register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to clear the task 8 interrupt overflow flag
6	INT7	R-0/W1S	0h	Writing a "1" to any of the bits will clear the corresponding MIOVF bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIOVF register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to clear the task 7 interrupt overflow flag
5	INT6	R-0/W1S	0h	Writing a "1" to any of the bits will clear the corresponding MIOVF bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIOVF register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to clear the task 6 interrupt overflow flag
4	INT5	R-0/W1S	0h	Writing a "1" to any of the bits will clear the corresponding MIOVF bit. Writes of "0" are ignored. Reads always return 0.  Notes: [1] Refer to MIOVF register description for handling of boundary conditions. Reset type: SYSRSn 0h (R/W) = This bit always reads back 0 and writes of 0 have no effect 1h (R/W) = Write a 1 to clear the task 5 interrupt overflow flag



**Table 5-56. MICLROVF Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	INT4	R-0/W1S	0h	<p>Writing a "1" to any of the bits will clear the corresponding MIOVF bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to clear the task 4 interrupt overflow flag</p>
2	INT3	R-0/W1S	0h	<p>Writing a "1" to any of the bits will clear the corresponding MIOVF bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to clear the task 3 interrupt overflow flag</p>
1	INT2	R-0/W1S	0h	<p>Writing a "1" to any of the bits will clear the corresponding MIOVF bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to clear the task 2 interrupt overflow flag</p>
0	INT1	R-0/W1S	0h	<p>Writing a "1" to any of the bits will clear the corresponding MIOVF bit. Writes of "0" are ignored. Reads always return 0.</p> <p>Notes: [1] Refer to MIOVF register description for handling of boundary conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = This bit always reads back 0 and writes of 0 have no effect</p> <p>1h (R/W) = Write a 1 to clear the task 1 interrupt overflow flag</p>

### 5.8.4.20 MIER Register (Offset = 25h) [reset = 0h]

MIER is shown in [Figure 5-29](#) and described in [Table 5-57](#).

Return to the [Summary Table](#).

Setting the bits in the interrupt enable register (MIER) allow an incoming interrupt or main CPU software to start the corresponding CLA task. Writing a 0 will block the task, but the interrupt request will still be latched in the flag register (MIFLG). Setting the MIER register bit to 0 while the corresponding task is executing will have no effect on the task. The task will continue to run until it hits the MSTOP instruction. When a soft reset is issued, the MIER bits are cleared. There should always be at least a 1 SYSCLKOUT delay between issuing the soft reset and reconfiguring the MIER bits.

**Figure 5-29. MIER Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
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 5-57. MIER Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R/W	0h	<p>Setting any of the bits to "1" enables the corresponding interrupt from triggering a corresponding CLA task. Writing a "0" blocks the interrupt, but the interrupt can still be latched by the MIFR register.</p> <p>When an interrupt is enabled and the corresponding MIFR bit is set to "1", the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.</p> <p>Interrupts are be serviced in normal priority order.</p> <p>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to "0", it will have no effect on the task. The task will run until it hits the STOP instruction.</p> <p>Reset type: SYRSn                      0h (R/W) = TASK_INT_DISABLE                      Task 8 interrupt is disabled (default)                      1h (R/W) = TASK_INT_ENABLE                      Task 8 interrupt is enabled</p>
6	INT7	R/W	0h	<p>Setting any of the bits to "1" enables the corresponding interrupt from triggering a corresponding CLA task. Writing a "0" blocks the interrupt, but the interrupt can still be latched by the MIFR register.</p> <p>When an interrupt is enabled and the corresponding MIFR bit is set to "1", the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.</p> <p>Interrupts are be serviced in normal priority order.</p> <p>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to "0", it will have no effect on the task. The task will run until it hits the STOP instruction.</p> <p>Reset type: SYRSn                      0h (R/W) = TASK_INT_DISABLE                      Task 7 interrupt is disabled (default)                      1h (R/W) = TASK_INT_ENABLE                      Task 7 interrupt is enabled</p>

**Table 5-57. MIER Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	INT6	R/W	0h	<p>Setting any of the bits to "1" enables the corresponding interrupt from triggering a corresponding CLA task. Writing a "0" blocks the interrupt, but the interrupt can still be latched by the MIFR register.</p> <p>When an interrupt is enabled and the corresponding MIFR bit is set to "1", the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.</p> <p>Interrupts are be serviced in normal priority order.</p> <p>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to "0", it will have no effect on the task. The task will run until it hits the STOP instruction.</p> <p>Reset type: SYSRSn            0h (R/W) = TASK_INT_DISABLE            Task 6 interrupt is disabled (default)            1h (R/W) = TASK_INT_ENABLE            Task 6 interrupt is enabled</p>
4	INT5	R/W	0h	<p>Setting any of the bits to "1" enables the corresponding interrupt from triggering a corresponding CLA task. Writing a "0" blocks the interrupt, but the interrupt can still be latched by the MIFR register.</p> <p>When an interrupt is enabled and the corresponding MIFR bit is set to "1", the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.</p> <p>Interrupts are be serviced in normal priority order.</p> <p>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to "0", it will have no effect on the task. The task will run until it hits the STOP instruction.</p> <p>Reset type: SYSRSn            0h (R/W) = TASK_INT_DISABLE            Task 5 interrupt is disabled (default)            1h (R/W) = TASK_INT_ENABLE            Task 5 interrupt is enabled</p>
3	INT4	R/W	0h	<p>Setting any of the bits to "1" enables the corresponding interrupt from triggering a corresponding CLA task. Writing a "0" blocks the interrupt, but the interrupt can still be latched by the MIFR register.</p> <p>When an interrupt is enabled and the corresponding MIFR bit is set to "1", the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.</p> <p>Interrupts are be serviced in normal priority order.</p> <p>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to "0", it will have no effect on the task. The task will run until it hits the STOP instruction.</p> <p>Reset type: SYSRSn            0h (R/W) = TASK_INT_DISABLE            Task 4 interrupt is disabled (default)            1h (R/W) = TASK_INT_ENABLE            Task 4 interrupt is enabled</p>

**Table 5-57. MIER Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INT3	R/W	0h	<p>Setting any of the bits to "1" enables the corresponding interrupt from triggering a corresponding CLA task. Writing a "0" blocks the interrupt, but the interrupt can still be latched by the MIFR register.</p> <p>When an interrupt is enabled and the corresponding MIFR bit is set to "1", the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.</p> <p>Interrupts are be serviced in normal priority order.</p> <p>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to "0", it will have no effect on the task. The task will run until it hits the STOP instruction.</p> <p>Reset type: SYSRSn                      0h (R/W) = TASK_INT_DISABLE                      Task 3 interrupt is disabled (default)                      1h (R/W) = TASK_INT_ENABLE                      Task 3 interrupt is enabled</p>
1	INT2	R/W	0h	<p>Setting any of the bits to "1" enables the corresponding interrupt from triggering a corresponding CLA task. Writing a "0" blocks the interrupt, but the interrupt can still be latched by the MIFR register.</p> <p>When an interrupt is enabled and the corresponding MIFR bit is set to "1", the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.</p> <p>Interrupts are be serviced in normal priority order.</p> <p>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to "0", it will have no effect on the task. The task will run until it hits the STOP instruction.</p> <p>Reset type: SYSRSn                      0h (R/W) = TASK_INT_DISABLE                      Task 2 interrupt is disabled (default)                      1h (R/W) = TASK_INT_ENABLE                      Task 2 interrupt is enabled</p>
0	INT1	R/W	0h	<p>Setting any of the bits to "1" enables the corresponding interrupt from triggering a corresponding CLA task. Writing a "0" blocks the interrupt, but the interrupt can still be latched by the MIFR register.</p> <p>When an interrupt is enabled and the corresponding MIFR bit is set to "1", the CLA will start executing the corresponding task and automatically clear the corresponding MIFR bit.</p> <p>Interrupts are be serviced in normal priority order.</p> <p>Notes: [1] If a task is currently executing and the corresponding MIER bit is cleared to "0", it will have no effect on the task. The task will run until it hits the STOP instruction.</p> <p>Reset type: SYSRSn                      0h (R/W) = TASK_INT_DISABLE                      Task 1 interrupt is disabled (default)                      1h (R/W) = TASK_INT_ENABLE                      Task 1 interrupt is enabled</p>

### 5.8.4.21 MIRUN Register (Offset = 26h) [reset = 0h]

MIRUN is shown in [Figure 5-30](#) and described in [Table 5-58](#).

Return to the [Summary Table](#).

The interrupt run status register (MIRUN) indicates which task is currently executing. Only one MIRUN bit will ever be set to a 1 at any given time. The bit is automatically cleared when the task completes and the respective interrupt is fed to the peripheral interrupt expansion (PIE) block of the device. This lets the main CPU know when a task has completed. The main CPU can stop a currently running task by writing to the MCTL[SOFTRESET] bit. This will clear the MIRUN flag and stop the task. In this case no interrupt will be generated to the PIE.

**Figure 5-30. MIRUN Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 5-58. MIRUN Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R	0h	<p>These bits indicate which task is currently active. Only one bit can be set to "1" at any one time. The bit is automatically cleared to "0" when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed.</p> <p>A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.</p> <p>Reset type: SYSRSn                      0h (R/W) = Task 8 is not executing (default)                      1h (R/W) = Task 8 is executing</p>
6	INT7	R	0h	<p>These bits indicate which task is currently active. Only one bit can be set to "1" at any one time. The bit is automatically cleared to "0" when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed.</p> <p>A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.</p> <p>Reset type: SYSRSn                      0h (R/W) = Task 7 is not executing (default)                      1h (R/W) = Task 7 is executing</p>
5	INT6	R	0h	<p>These bits indicate which task is currently active. Only one bit can be set to "1" at any one time. The bit is automatically cleared to "0" when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed.</p> <p>A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.</p> <p>Reset type: SYSRSn                      0h (R/W) = Task 6 is not executing (default)                      1h (R/W) = Task 6 is executing</p>

**Table 5-58. MIRUN Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	INT5	R	0h	<p>These bits indicate which task is currently active. Only one bit can be set to "1" at any one time. The bit is automatically cleared to "0" when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed.</p> <p>A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.</p> <p>Reset type: SYSRSn                      0h (R/W) = Task 5 is not executing (default)                      1h (R/W) = Task 5 is executing</p>
3	INT4	R	0h	<p>These bits indicate which task is currently active. Only one bit can be set to "1" at any one time. The bit is automatically cleared to "0" when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed.</p> <p>A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.</p> <p>Reset type: SYSRSn                      0h (R/W) = Task 4 is not executing (default)                      1h (R/W) = Task 4 is executing</p>
2	INT3	R	0h	<p>These bits indicate which task is currently active. Only one bit can be set to "1" at any one time. The bit is automatically cleared to "0" when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed.</p> <p>A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.</p> <p>Reset type: SYSRSn                      0h (R/W) = Task 3 is not executing (default)                      1h (R/W) = Task 3 is executing</p>
1	INT2	R	0h	<p>These bits indicate which task is currently active. Only one bit can be set to "1" at any one time. The bit is automatically cleared to "0" when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed.</p> <p>A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.</p> <p>Reset type: SYSRSn                      0h (R/W) = Task 2 is not executing (default)                      1h (R/W) = Task 2 is executing</p>

**Table 5-58. MIRUN Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	INT1	R	0h	<p>These bits indicate which task is currently active. Only one bit can be set to "1" at any one time. The bit is automatically cleared to "0" when the task completes and the respective CLAINTxn line is toggled to indicate task completion. The CLAINTxn interrupt line can be fed to the PIE of the CPU so the CPU knows when a task has completed.</p> <p>A currently running task can be stopped by a SOFTRESET. The RUN flag is cleared, the task is stopped, but no CLAINTxn interrupt is generated.</p> <p>Reset type: SYSRSn            0h (R/W) = Task 1 is not executing (default)            1h (R/W) = Task 1 is executing</p>

### 5.8.4.22 \_MPC Register (Offset = 28h) [reset = 0h]

\_MPC is shown in [Figure 5-31](#) and described in [Table 5-59](#).

Return to the [Summary Table](#).

CLA Program Counter

**Figure 5-31. \_MPC Register**

15	14	13	12	11	10	9	8
_MPC							
R-0h							
7	6	5	4	3	2	1	0
_MPC							
R-0h							

**Table 5-59. \_MPC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	_MPC	R	0h	<p>Program Counter: The PC value is initialized by the appropriate MVECTx register when an interrupt (task) is serviced.</p> <p>The MPC register address 16-bits and not 32-bits. Hence the address range of the CLA with a 16-bit MPC is 64Kx16 words or 32K CLA instructions.</p> <p>Notes: [1] To be consistent with C28 core implementation, the PC value points to the instruction in D2 stage of pipeline.</p> <p>[2] After a STOP operation, and with no other task pending, the PC will remain pointing to the STOP operation.</p> <p>Reset type: SYSRSn</p>



**5.8.4.23 \_MAR0 Register (Offset = 2Ah) [reset = 0h]**

\_MAR0 is shown in [Figure 5-32](#) and described in [Table 5-60](#).

Return to the [Summary Table](#).

CLA Auxiliary Register 0

**Figure 5-32. \_MAR0 Register**

15	14	13	12	11	10	9	8
_MAR0							
R-0h							
7	6	5	4	3	2	1	0
_MAR0							
R-0h							

**Table 5-60. \_MAR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	_MAR0	R	0h	CLA Auxillary Register 0 Reset type: SYSRSn

**5.8.4.24 \_MAR1 Register (Offset = 2Bh) [reset = 0h]**

\_MAR1 is shown in [Figure 5-33](#) and described in [Table 5-61](#).

Return to the [Summary Table](#).

CLA Auxiliary Register 1

**Figure 5-33. \_MAR1 Register**

15	14	13	12	11	10	9	8
_MAR1							
R-0h							
7	6	5	4	3	2	1	0
_MAR1							
R-0h							

**Table 5-61. \_MAR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	_MAR1	R	0h	CLA Auxillary Register 1 Reset type: SYSRSn

### 5.8.4.25 \_MSTF Register (Offset = 2Eh) [reset = 0h]

\_MSTF is shown in [Figure 5-34](#) and described in [Table 5-62](#).

Return to the [Summary Table](#).

The CLA status register (MSTF) reflects the results of different operations. These are the basic rules for the flags:

- Zero and negative flags are cleared or set based on:
  - floating-point moves to registers
  - the result of compare, minimum, maximum, negative and absolute value operations
  - the integer result of operations such as MMOV16, MAND32, MOR32, MXOR32, MCMP32, MASR32, MLSR32
- Overflow and underflow flags are set by floating-point math instructions such as multiply, add, subtract and 1/x. These flags may also be connected to the peripheral interrupt expansion (PIE) block on your device. This can be useful for debugging underflow and overflow conditions within an application.

**Figure 5-34. \_MSTF Register**

31	30	29	28	27	26	25	24
RESERVED				_RPC			
R-0h				R-0h			
23	22	21	20	19	18	17	16
_RPC							
R-0h							
15	14	13	12	11	10	9	8
_RPC				MEALLOW	RESERVED	RNDF32	RESERVED
R-0h				R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	TF	RESERVED		ZF	NF	LUF	LVF
R-0h	R-0h	R-0h		R-0h	R-0h	R-0h	R-0h

**Table 5-62. \_MSTF Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-12	_RPC	R	0h	Return program counter The _RPC is used to save and restore the MPC address by the MCCNDD and MRCNDD operations Reset type: SYSRSn
11	MEALLOW	R	0h	MEALLOW Status This bit enables and disables CLA write access to EALLOW protected registers This is independent of the state of the EALLOW bit in the main CPU status register This status bit can be saved and restored by the MMOV32 STF, mem32 instruction Reset type: SYSRSn 0h (R/W) = The CLA cannot write to EALLOW protected registers. This bit is cleared by the CLA instruction, MEDIS. 1h (R/W) = The CLA is allowed to write to EALLOW protected registers. This bit is set by the CLA instruction, MEALLOW.
10	RESERVED	R	0h	Reserved
9	RNDF32	R	0h	Round 32-bit Floating-Point Mode Use the MSETFLG and MMOV32 MSTF, mem32 instructions to change the rounding mode Reset type: SYSRSn 0h (R/W) = If this bit is zero, the MMPYF32, MADD32 and MSUBF32 instructions will round to zero (truncate). 1h (R/W) = If this bit is one, the MMPYF32, MADD32 and MSUBF32 instructions will round to the nearest even value.

**Table 5-62. \_MSTF Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-7	RESERVED	R	0h	Reserved
6	TF	R	0h	Test Flag The MTESTTF instruction can modify this flag based on the condition tested. The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag. Reset type: SYSRSn 0h (R/W) = The condition tested with the MTESTTF instruction is false. 1h (R/W) = The condition tested with the MTESTTF instruction is true.
5-4	RESERVED	R	0h	Reserved
3	ZF	R	0h	Zero Flag - Instructions that modify this flag based on the floating-point value stored in the destination register: MMOV32, MMOVD32, MABSF32, MNEGF32 - Instructions that modify this flag based on the floating-point result of the operation: MCMPF32, MMAXF32, and MMINF32 - Instructions that modify this flag based on the integer result of the operation: MMOV16, MAND32, MOR32, MXOR32, MCMP32, MASR32, MLSR32 and MLSL32 The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag. Reset type: SYSRSn 0h (R/W) = The value is not zero 1h (R/W) = The value is zero
2	NF	R	0h	Negative Flag - Instructions that modify this flag based on the floating-point value stored in the destination register: MMOV32, MMOVD32, MABSF32, MNEGF32 - Instructions that modify this flag based on the floating-point result of the operation: MCMPF32, MMAXF32, and MMINF32 - Instructions that modify this flag based on the integer result of the operation: MMOV16, MAND32, MOR32, MXOR32, MCMP32, MASR32, MLSR32 and MLSL32 The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag. Reset type: SYSRSn 0h (R/W) = The value is not negative 1h (R/W) = The value is negative
1	LUF	R	0h	Latched Underflow Flag The following instructions will set this flag to 1 if an underflow occurs: MPPYF32, MADDF32, MSUBF32, MMACF32, MEINVF32, MEISQRTF32 The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag. Reset type: SYSRSn 0h (R/W) = An underflow condition has not been latched 1h (R/W) = An underflow condition has been latched

**Table 5-62. \_MSTF Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	LVF	R	0h	<p>Latched Overflow Flag</p> <p>The following instructions will set this flag to 1 if an overflow occurs: MMPYF32, MADDF32, MSUBF32, MMACF32, MEINVF32, MEISQRTF32</p> <p>The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = An overflow condition has not been latched</p> <p>1h (R/W) = An overflow condition has been latched</p>

### 5.8.4.26 \_MR0 Register (Offset = 30h) [reset = 0h]

\_MR0 is shown in [Figure 5-35](#) and described in [Table 5-63](#).

Return to the [Summary Table](#).

CLA Floating-Point Result Register 0

**Figure 5-35. \_MR0 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
																	i32														
																	R-0h														

**Table 5-63. \_MR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	i32	R	0h	CLA Result Register 0 Reset type: SYSRSn

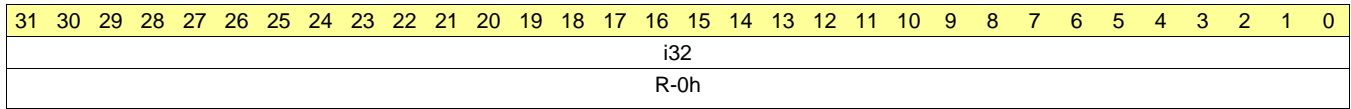
**5.8.4.27 \_MR1 Register (Offset = 34h) [reset = 0h]**

\_MR1 is shown in [Figure 5-36](#) and described in [Table 5-64](#).

Return to the [Summary Table](#).

CLA Floating-Point Result Register 1

**Figure 5-36. \_MR1 Register**



**Table 5-64. \_MR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	i32	R	0h	CLA Result Register 1 Reset type: SYSRSn

**5.8.4.28 \_MR2 Register (Offset = 38h) [reset = 0h]**

\_MR2 is shown in [Figure 5-37](#) and described in [Table 5-65](#).

Return to the [Summary Table](#).

CLA Floating-Point Result Register 2

**Figure 5-37. \_MR2 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
																	i32														
																	R-0h														

**Table 5-65. \_MR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	i32	R	0h	CLA Result Register 2 Reset type: SYSRSn



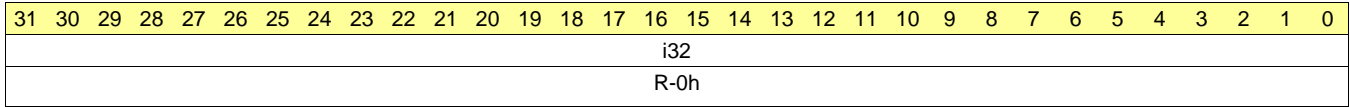
**5.8.4.29 \_MR3 Register (Offset = 3Ch) [reset = 0h]**

\_MR3 is shown in [Figure 5-38](#) and described in [Table 5-66](#).

Return to the [Summary Table](#).

CLA Floating-Point Result Register 3

**Figure 5-38. \_MR3 Register**



**Table 5-66. \_MR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	i32	R	0h	CLA Result Register 3 Reset type: SYSRSn

**5.8.4.30 \_MPSACTL Register (Offset = 42h) [reset = 0h]**

\_MPSACTL is shown in [Figure 5-39](#) and described in [Table 5-67](#).

Return to the [Summary Table](#).

PSA Control Register

**Figure 5-39. \_MPSACTL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
MPSA2CFG		MPSA2CLEAR	MPSA1CLEAR	MDWDBCYC	MDWDBSTAR T	MPABCYC	MPABSTART
R/W-0h		R-0/W1S-0h	R-0/W1S-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 5-67. \_MPSACTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-6	MPSA2CFG	R/W	0h	CLA PSA2 Polynomial Configuration Bits: These bits configure the type of polynomial used for PSA2. The polynomials chosen are commonly used in the industry: Mode Polynomial Type 0,0 PSA 0,1 CRC32 1,0 CRC16 1,1 CRC16-CCITT  Note: [1] Polynomial configuration should be performed when PSA2 is stopped. Reset type: SYSRSn
5	MPSA2CLEAR	R-0/W1S	0h	CLA PSA2 Clear Bit: Writing of "1" will clear contents of PSA2 register. Writes of "0" are ignored. Always reads back a "0"  Note: Clearing operation should be performed when PSA2 is stopped. Reset type: SYSRSn
4	MPSA1CLEAR	R-0/W1S	0h	CLA PSA1 Clear Bit: Writing of "1" will clear contents of PSA1 register. Writes of "0" are ignored. Always reads back a "0"  Note: Clearing operation should be performed when PSA1 is stopped. Reset type: SYSRSn
3	MDWDBCYC	R/W	0h	CLA Data Write Data Bus PSA2 Cycle or Event Based Bit: 0 PSA2 calculated on every cycle 1 PSA2 calculated on every bus event Reset type: SYSRSn
2	MDWDBSTART	R/W	0h	CLA Data Write Data Bus PSA2 Start/Stop Bit: 0 PSA2 stopped 1 PSA2 start Reset type: SYSRSn

**Table 5-67. \_MPSACTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	MPABCYC	R/W	0h	CLA Program Address Bus PSA1 Cycle/Event Based Bit: 0 PSA1 calculated on every cycle 1 PSA1 calculated on every bus event Reset type: SYSRSn
0	MPABSTART	R/W	0h	CLA Program Address Bus PSA1 Start/Stop Bit: 0 PSA1 stopped 1 PSA1 start Reset type: SYSRSn

**5.8.4.31 \_MPSA1 Register (Offset = 44h) [reset = 0h]**

\_MPSA1 is shown in [Figure 5-40](#) and described in [Table 5-68](#).

Return to the [Summary Table](#).

PSA1 Register

**Figure 5-40. \_MPSA1 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
i32																															
R/W-0h																															

**Table 5-68. \_MPSA1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	i32	R/W	0h	PSA1 Value: Reading this register gives the current PSA1 value. The value can be read at any time.  Writes to this register are allowed to initialize the PSA1 to a known value. Writes to this register should only be made when PSA1 is stopped.  Register value is cleared to zero by reset or by writing to the MPSA1CLEAR bit in the MPSACTL register. Reset type: SYSRSn

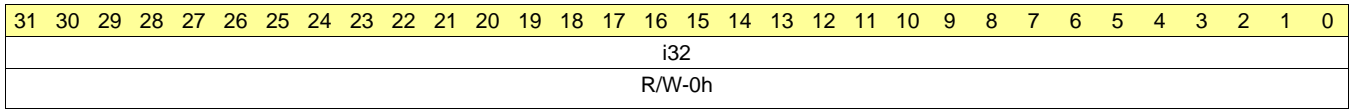
**5.8.4.32 \_MPSA2 Register (Offset = 46h) [reset = 0h]**

\_MPSA2 is shown in [Figure 5-41](#) and described in [Table 5-69](#).

Return to the [Summary Table](#).

PSA2 Register

**Figure 5-41. \_MPSA2 Register**



**Table 5-69. \_MPSA2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	i32	R/W	0h	PSA2 Value: Reading this register gives the current PSA2 value. The value can be read at any time.  Writes to this register are allowed to initialize the PSA2 to a known value. Writes to this register should only be made when PSA2 is stopped.  Register value is cleared to zero by reset or by writing to the MPSA2CLEAR bit in the MPSACTL register. Reset type: SYSRSn

## Dual-Clock Comparator (DCC)

---

---

This chapter describes the Dual-Clock Comparator (DCC) module.

Topic	Page
<b>6.1 Introduction</b> .....	<b>847</b>
<b>6.2 Features</b> .....	<b>847</b>
<b>6.3 Module Operation</b> .....	<b>847</b>
<b>6.4 Interrupts</b> .....	<b>849</b>
<b>6.5 DCC Registers</b> .....	<b>850</b>

## 6.1 Introduction

The dual-clock comparator module is used for evaluating and monitoring the clock input based on a second clock, which can be a more accurate and reliable version. This instrumentation is used to detect faults in clock source or clock structures, thereby enhancing the system's safety metrics.

## 6.2 Features

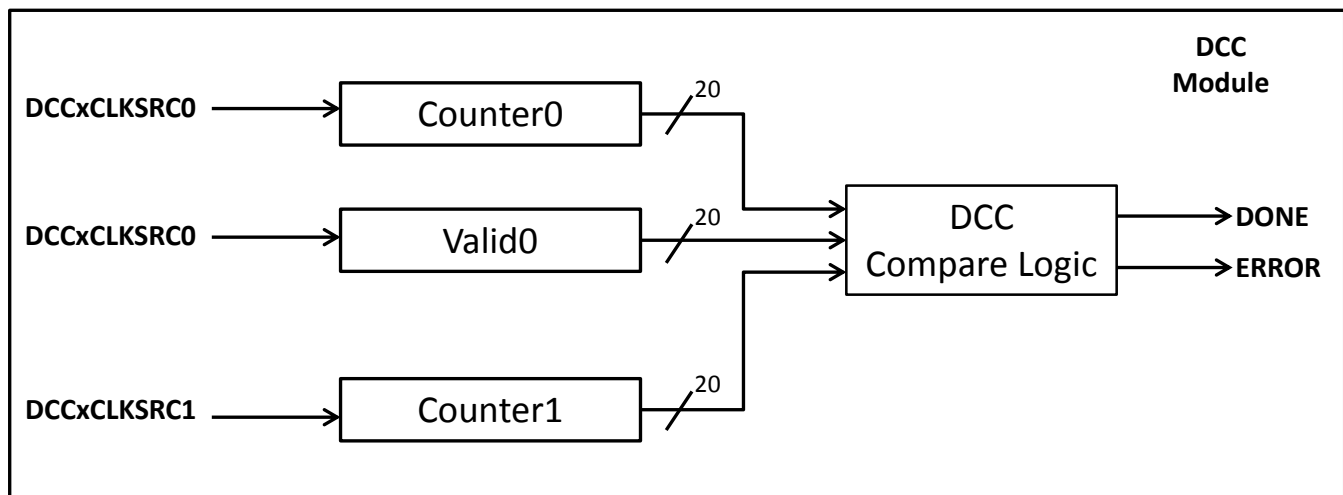
The main features of each of the DCC modules are:

- Allows the application to ensure that a fixed ratio is maintained between frequencies of two clock signals.
- Supports the definition of a programmable tolerance window in terms of the number of reference clock cycles.
- Supports a single-sequence mode for spot measurements.
- Allows the selection of a clock source for each of the counters, resulting in several specific use cases.

### 6.2.1 Block Diagram

The DCC block diagram illustrates the main concept of the DCC module.

**Figure 6-1. DCC Operation**



## 6.3 Module Operation

As shown in [Figure 6-1](#), DCC contains three counters – Counter0, Valid0 and Counter1. Initially, all counters are loaded with their user-defined, pre-load value. Counter0 and Counter1 start decrementing once the DCC is enabled at rates determined by the frequencies of Clock0 and Clock1, respectively. When Counter0 equals 0 (expires), the Valid0 counter begins decrementing at a rate determined by Clock0. If Counter1 decrements to 0 in the valid window, then no error is generated and Clock1 is considered to be good within allowable tolerance as configured by the user.

### 6.3.1 Configuring DCC Counters

Counter0 and Counter1 are configured based on the ratio between the frequencies of Clock0 and Clock1 ( $F_{clk1} \times \text{Counter0} = F_{clk0} \times \text{Counter1}$ ). The Valid0 counter provides tolerance and is configured based on the error in DCC. Since Clock0 and Clock1 are asynchronous, the start and stop of the counters do not occur synchronously. Hence, while configuring the counters, two different sources of errors must be accounted for:

- Errors due to the asynchronous timing of Clock0 and Clock1 - This depends on the frequency of Clock0 and Clock1:
  - If  $F_{clk1} > F_{clk0}$ , then Async. Error (In Clock0 cycles) = 2

- If  $F_{clk1} < F_{clk0}$ , then Async. Error (In Clock0 cycles) =  $2 \times (F_{clk0}/F_{clk1})$
- Digitization error - 3 Clock0 cycles

**Total Error (in Clock0 Cycles) = Async. Error + Digitization Error**

Another consideration while configuring the counters is the amount of frequency tolerance that users can accept in the frequency of the clock under measurement. If the desired tolerance is low, then the counter values will be large and will increase the window of measurement. This means that counter values for a tolerance of 0.1% will be larger than that of 0.2%. So, based on the tolerance, users can define the window of measurement in terms of Clock0 cycles. To illustrate:

**Window (in Clock0 Cycles) = (Total Error) / (0.01 × Tolerance)**

For example, if the total Error is 10 and the tolerance desired is +/-0.1%, then:

**Window (In Clock0 cycles) =  $10 / (0.01 \times 0.1) = 10000$**

The following equations should be used to configure counter values:

**Counter0 = Window - Total Error**

**Valid0 = 2 × Total Error**

**Counter1 = Window × (Fclk1/Fclk0)**

### 6.3.2 Single-Shot Measurement Mode

The DCC module can be programmed to count down one time by enabling the single-shot mode. In this mode, the DCC stops operating when the down counter0 and the valid counter0 reach 0.

At the end of one sequence of counting down in this single-shot mode, the DCC gets disabled automatically, which prevents further counting. This mode is typically used for spot checking the frequency of a signal.

**Example-1: Validating PLLRAWCLK frequency**

A practical example of the usage is to validate the PLL output clock frequency using the XTAL as the reference clock. Assume XTAL is 10Mhz, PLL output frequency is 100Mhz and tolerance required is 0.1%. The measurement sequence would proceed as follows:

- Set Clock0 source for Counter0 and Valid0 as XTAL, and Clock1 source for Counter1 as PLL output clock.
- Based on the equations defined in [Section 6.3.1](#), calculated seed values for Counters will be: Counter0 = 4995; Valid0 = 10; Counter1 = 50000
- Once the DCC is enabled, the counters Counter0 and Counter1 both start counting down from their seed values.
- When Counter0 reaches zero, it automatically triggers the Valid0 counter.
- When Valid0 reaches zero and Counter1 is not zero, an ERROR status flag is set and a "DCC error" is sent to the PIE. Counter1 is frozen so that it stops counting down any further. The application can enable an interrupt to be generated from the PIE whenever this DCC error is indicated. Refer to [Table 3-3](#) to know the channel mapping of DCC Interrupt.
- The application then needs to clear the ERROR status flag and restart the DCC module so that it is ready for the next spot measurement.

If there is no error generated at the end of the sequence, then the DONE status flag is set and a DONE interrupt is generated. The application must clear the DONE flag before restarting the DCC.

**Error Conditions:**

An error condition is generated by any one of the following:

1. Counter1 counts down to 0 before Counter0 reaches 0. This means that Clock1 is faster than expected, or Clock0 is slower than expected. It includes the case when Clock0 is stuck at 1 or 0.
2. Counter1 does not reach 0 even when Counter0 and Valid0 have both reached 0. This means that Clock1 is slower than expected. It includes the case when Clock1 is stuck at 1 or 0.



Any error freezes the counters from counting. An application may then read out the counter values to help determine what caused the error.

## 6.4 Interrupts

DCC generates interrupt on either of the two events:

- DCC finishes counting and all the counters expire within defined window indicating DONE operation, provided `DCCGCTRL.DONENA=1`.
- DCC finishes counting with error where counters do not expire in defined window. This indicates an ERROR event, and sets an interrupt provided `DCCGCTRL.ERRENA=1`.

Interrupt due to DONE or ERROR event is OR'd and is flagged as DCC, which goes to INT7.16 in PIE Interrupt table. Application ISR needs to check the status flag inside `DCCSTATUS` register to determine whether it is due to ERROR or DONE.

## 6.5 DCC Registers

This section describes the Dual Clock Comparator Registers.

### 6.5.1 DCC Base Addresses

**Table 6-1. DCC Base Address Table**

Device Registers	Register Name	Start Address	End Address
DCC0Regs	DCC_REGS	0x0005_E700	0x0005_E71C

## 6.5.2 DCC\_REGS Registers

Table 6-2 lists the DCC\_REGS registers. All register offset addresses not listed in Table 6-2 should be considered as reserved locations and the register contents should not be modified.

**Table 6-2. DCC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	DCCGCTRL	Global Control Register		<a href="#">Go</a>
4h	DCCREV	DCC Revision Register		<a href="#">Go</a>
8h	DCCCNTSEED0	Counter 0 Seed Value		<a href="#">Go</a>
Ch	DCCVALIDSEED0	Valid 0 Seed Value		<a href="#">Go</a>
10h	DCCCNTSEED1	Counter 1 Seed Value		<a href="#">Go</a>
14h	DCCSTATUS	DCC Status		<a href="#">Go</a>
18h	DCCCNT0	Counter 0 Value		<a href="#">Go</a>
1Ch	DCCVALID0	Valid Value 0		<a href="#">Go</a>
20h	DCCCNT1	Counter 1 Value		<a href="#">Go</a>
24h	DCCCLKSRC1	Clock Source 1		<a href="#">Go</a>
28h	DCCCLKSRC0	Clock Source 0		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 6-3 shows the codes that are used for access types in this section.

**Table 6-3. DCC\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R -0	Read Returns 0s
R-1	R -1	Read Returns 1s
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 6.5.2.1 DCCGCTRL Register (Offset = 0h) [reset = 5555h]

DCCGCTRL is shown in [Figure 6-2](#) and described in [Table 6-4](#).

Return to the [Summary Table](#).

Starts / stops the counters. Clears the error signal.

**Figure 6-2. DCCGCTRL 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
DONEENA				SINGLESHOT				ERRENA				DCCENA			
R/W-5h				R/W-5h				R/W-5h				R/W-5h			

**Table 6-4. DCCGCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W	0h	Reserved
15-12	DONEENA	R/W	5h	<p>DONE Enable</p> <p>Enables/disables the done interrupt signal, but has no effect on the done status flag in DCCSTAT register.</p> <p>0101 The done signal is disabled</p> <p>Others The done signal is enabled</p> <p>Reset type: SYSRSn</p>
11-8	SINGLESHOT	R/W	5h	<p>Single-Shot Enable</p> <p>Enables/disables repetitive operation of the DCC.</p> <p>1010 Stop counting when COUNTER0 and VALID0 both reach zero</p> <p>Note: Configure this to 0xA before Enabling DCC</p> <p>Note: All values other than 1010 are reserved</p> <p>Reset type: SYSRSn</p>
7-4	ERRENA	R/W	5h	<p>Error Enable</p> <p>Enables/disables the error signal.</p> <p>0101 The error signal is disabled</p> <p>Others The error signal is enabled</p> <p>Reset type: SYSRSn</p>
3-0	DCCENA	R/W	5h	<p>DCC Enable</p> <p>Starts and stops the operation of the DCC.</p> <p>0101 Counters are stopped</p> <p>Others Counters are running</p> <p>Reset type: SYSRSn</p>

### 6.5.2.2 DCCREV Register (Offset = 4h) [reset = 40041003h]

DCCREV is shown in [Figure 6-3](#) and described in [Table 6-5](#).

Return to the [Summary Table](#).

Specifies the module version.

**Figure 6-3. DCCREV Register**

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

**Table 6-5. DCCREV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R-1	1h	Reserved
29-28	RESERVED	R-0	0h	Reserved
27-16	RESERVED	R	4h	Reserved
15-11	RESERVED	R	2h	Reserved
10-8	MAJOR	R	0h	Major Revision Number Represents major changes to the module (e.g. entirely new features are added/changed). The major revision number for this module. Reset type: SYSRSn
7-6	RESERVED	R	0h	Reserved
5-0	MINOR	R	3h	Minor Revision Number Represents minor changes to the module (e.g. enhancements to existing features). The minor revision number for this module. Reset type: SYSRSn

### 6.5.2.3 DCCNTSEED0 Register (Offset = 8h) [reset = 0h]

DCCNTSEED0 is shown in [Figure 6-4](#) and described in [Table 6-6](#).

Return to the [Summary Table](#).

Seed value for the counter attached to Clock Source 0.

**Figure 6-4. DCCNTSEED0 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												COUNTSEED0																			
R-0h												R/W-0h																			

**Table 6-6. DCCNTSEED0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-0	COUNTSEED0	R/W	0h	Seed Value for Counter 0 Contains the seed value that gets loaded into Counter 0 (Clock Source 0).  NOTE: Operating the DCC with '0' in the COUNTSEED0 register will result in undefined operation. Reset type: SYSRSn

#### 6.5.2.4 DCCVALIDSEED0 Register (Offset = Ch) [reset = 0h]

DCCVALIDSEED0 is shown in [Figure 6-5](#) and described in [Table 6-7](#).

Return to the [Summary Table](#).

Seed value for the timeout counter attached to Clock Source 0.

**Figure 6-5. DCCVALIDSEED0 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																VALIDSEED															
R-0h																R/W-0h															

**Table 6-7. DCCVALIDSEED0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALIDSEED	R/W	0h	Seed Value for Valid Duration Counter 0 Contains the seed value that gets loaded into the valid duration counter for Clock Source 0.  NOTE: Operating the DCC with '0' in the VALIDSEED0 register will result in undefined operation. VALID0 defines a window in which COUNT1 expires. This window is meant to be at least four cycles wide. Do not program a value less than '4' into the VALID0 register. Reset type: SYSRSn

### 6.5.2.5 DCCNTSEED1 Register (Offset = 10h) [reset = 0h]

DCCNTSEED1 is shown in [Figure 6-6](#) and described in [Table 6-8](#).

Return to the [Summary Table](#).

Seed value for the counter attached to Clock Source 1.

**Figure 6-6. DCCNTSEED1 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												COUNTSEED1																			
R-0h												R/W-0h																			

**Table 6-8. DCCNTSEED1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-0	COUNTSEED1	R/W	0h	Seed Value for Counter 1 Contains the seed value that gets loaded into Counter 1 (Clock Source 1).  NOTE: Operating the DCC with '0' in the COUNTSEED1 register will result in undefined operation. Reset type: SYSRSn



**6.5.2.6 DCCSTATUS Register (Offset = 14h) [reset = 0h]**

DCCSTATUS is shown in [Figure 6-7](#) and described in [Table 6-9](#).

Return to the [Summary Table](#).

Specifies the status of the DCC Module.

**Figure 6-7. DCCSTATUS 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						DONE	ERR
R-0h						R/W-0h	R/W-0h

**Table 6-9. DCCSTATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	DONE	R/W	0h	Single-Shot Done Flag Indicates when single-shot mode is complete without error. Writing a '1' to this bit clears the flag.  0 Single-shot mode has not completed. 1 Single-shot mode has completed. Reset type: SYSRSn
0	ERR	R/W	0h	Error Flag Indicates whether or not an error has occurred. Writing a '1' to this bit clears the flag.  0 No errors have occurred. 1 An error has occurred. Reset type: SYSRSn

### 6.5.2.7 DCCNT0 Register (Offset = 18h) [reset = 0h]

DCCNT0 is shown in [Figure 6-8](#) and described in [Table 6-10](#).

Return to the [Summary Table](#).

Value of the counter attached to Clock Source 0.

**Figure 6-8. DCCNT0 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												COUNT0																			
R-0h												R-0h																			

**Table 6-10. DCCNT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-0	COUNT0	R	0h	Current Value of Counter 0 Reset type: SYSRSn

### 6.5.2.8 DCCVALID0 Register (Offset = 1Ch) [reset = 0h]

DCCVALID0 is shown in [Figure 6-9](#) and described in [Table 6-11](#).

Return to the [Summary Table](#).

Value of the valid counter attached to Clock Source 0.

**Figure 6-9. DCCVALID0 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																VALID0															
R-0h																R-0h															

**Table 6-11. DCCVALID0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALID0	R	0h	Current Value of Valid 0 Reset type: SYSRSn

### 6.5.2.9 DCCNT1 Register (Offset = 20h) [reset = 0h]

DCCNT1 is shown in [Figure 6-10](#) and described in [Table 6-12](#).

Return to the [Summary Table](#).

Value of the counter attached to Clock Source 1.

**Figure 6-10. DCCNT1 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												COUNT1																			
R-0h												R-0h																			

**Table 6-12. DCCNT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-0	COUNT1	R	0h	Current Value of Counter 1 Reset type: SYSRSn

### 6.5.2.10 DCCCLKSRC1 Register (Offset = 24h) [reset = 5002h]

DCCCLKSRC1 is shown in [Figure 6-11](#) and described in [Table 6-13](#).

Return to the [Summary Table](#).

Selects the clock source for Counter 1.

**Figure 6-11. DCCCLKSRC1 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
KEY				RESERVED								CLKSRC1			
R-0/W-5h				R-0h								R/W-2h			

**Table 6-13. DCCCLKSRC1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	KEY	R-0/W	5h	Enables or Disables Clock Source Write for COUNT1  1010: The CLKSRC1 field written with key gets updated to with new selection to clock COUNT1.  Others: Previous values retained new writes on register fields has no impact.  Reset type: SYSRSn
11-4	RESERVED	R	0h	Reserved
3-0	CLKSRC1	R/W	2h	Clock Source Select for Counter 1 Specifies the clock source for COUNT1, when the KEY field enables this feature.  0000 Clock Source 0 (PLLRAWCLK) is selected for COUNT1.  Note: Configure this value to 0x0 before enabling DCC Note: All values other than 0000 are reserved.  Reset type: SYSRSn

### 6.5.2.11 DCCCLKSRC0 Register (Offset = 28h) [reset = 1h]

DCCCLKSRC0 is shown in [Figure 6-12](#) and described in [Table 6-14](#).

Return to the [Summary Table](#).

Selects the clock source for Counter 0.

**Figure 6-12. DCCCLKSRC0 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												CLKSRC0			
R-0h												R/W-1h			

**Table 6-14. DCCCLKSRC0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	CLKSRC0	R/W	1h	<p>Clock Source Select for Counter 0</p> <p>0000 Clock Source 0(XTAL) is selected for COUNT0.  0001 Clock Source 1(INTOSC1) is selected for COUNT0.  0010 Clock Source 2(INTOSC2) is selected for COUNT0.</p> <p>Note: Reference Clock for the PLL (OSCCLKSRCSEL) and the DCCCLKSRC0 selected should be the same, meaning if OSCCLKSRCSEL=0x1 (XTAL) then DCCCLKSRC0.CLKRSRC0=0x0 (XTAL).</p> <p>Note: All other values are reserved.</p> <p>Reset type: SYSRSn</p>

### 6.5.3 Register to Driverlib Function Mapping

**Table 6-15. DCC Registers to Driverlib Functions**

File	Driverlib Function
<b>GCTRL</b>	
dcc.h	DCC_enableModule
dcc.h	DCC_disableModule
dcc.h	DCC_enableErrorSignal
dcc.h	DCC_enableDoneSignal
dcc.h	DCC_disableErrorSignal
dcc.h	DCC_disableDoneSignal
dcc.h	DCC_enableSingleShotMode
dcc.h	DCC_disableSingleShotMode
<b>REV</b>	
dcc.c	DCC_getRevisionNumber
<b>CNTSEED0</b>	
dcc.h	DCC_setCounterSeeds
<b>VALIDSEED0</b>	
dcc.h	DCC_setCounterSeeds
<b>CNTSEED1</b>	
dcc.h	DCC_setCounterSeeds
<b>STATUS</b>	
dcc.h	DCC_getErrorStatus
dcc.h	DCC_getSingleShotStatus
dcc.h	DCC_clearErrorFlag
dcc.h	DCC_clearDoneFlag
sysctl.c	SysCtl_isPLLValid
<b>CNT0</b>	
dcc.h	DCC_getCounter0Value
<b>VALID0</b>	
dcc.h	DCC_getValidCounter0Value
<b>CNT1</b>	
dcc.h	DCC_getCounter1Value
<b>CLKSRC1</b>	
dcc.h	DCC_setCounter1ClkSource
dcc.h	DCC_getCounter1ClkSource
<b>CLKSRC0</b>	
dcc.h	DCC_setCounter0ClkSource
dcc.h	DCC_getCounter0ClkSource

## **CLA Program ROM CRC (CLAPROMCRC)**

---

---

The CLAPROMCRC is a feature which calculates a CRC-32 value of a configurable block of data in the Control Law Accelerator program ROM space.

<b>Topic</b>	<b>Page</b>
<b>7.1 Overview .....</b>	<b>865</b>
<b>7.2 Functional Description .....</b>	<b>865</b>
<b>7.3 CLAPROM Registers .....</b>	<b>867</b>



## 7.1 Overview

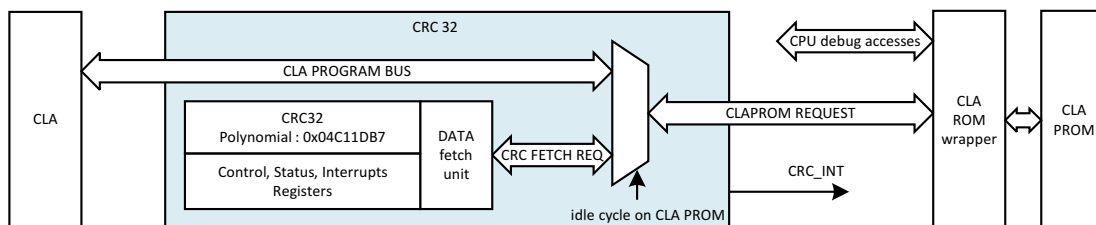
The CLAPROMCRC is a feature which calculates a CRC-32 on a configurable block of memory in the CLA program ROM space. Neither the C28x nor the CLA have the capability to compute a CRC on the CLA program ROM. The CLAPROMCRC solves this problem by calculating a CRC-32 in a non-intrusive manner. It is a hardware CRC-32 module which fetches the CLA program ROM during idle cycles (when the CLA is not accessing the ROM on the CLA program bus), and calculates the CRC-32 in order to perform a code integrity check. It then compares the result with a golden CRC-32 value and indicates a pass or fail condition.

## 7.2 Functional Description

The CLAPROMCRC module can be configured and initiated by either the C28x or the CLA. Once the CLAPROMCRC is initiated, the processing core can switch back to the application and service the CLAPROMCRC via interrupt. An interrupt can trigger the C28x and CLA and signal that the CRC-32 calculation and comparison has completed (CRCDONE). The CLAPROMCRC module waits for CLA program bus idle cycles in order to access the CLA program ROM. This ensures that the CRC-32 hardware calculation does not affect code execution and performance on the CLA.

Below is a functional diagram of the CLAPROMCRC module.

**Figure 7-1. CLAPROMCRC Functional Diagram**



Once the CLAPROMCRC is initiated by the C28x or CLA, the module will snoop for idle cycles to fetch program memory and calculate the CRC-32. It will begin fetching from the START\_ADDRESS and stop after the CRC-32 has been calculated for the required number of bytes specified by BLOCK\_SIZE. The CRC-32 can be calculated on as little as 1KB of data up to the size of the CLA program ROM in increments of 1KB.

The CLAPROMCRC module fetches 32 bits of program data on every fetch and takes four cycles to calculate the CRC-32 value for each fetch. For example, in order to calculate a CRC-32 on 1KB of program data, the module requires 256 idle cycles if each fetch access is spaced four cycles apart. Therefore, to calculate the CRC-32 on a 1KB block of program data requires a minimum of 1024 cycles. The CRC polynomial used in the hardware calculation is 0x04C11DB7.

### 7.2.1 Start Address

The start address must be aligned to a 1KB boundary (512 word boundary). If a non-aligned address is programmed, then the 9 least significant bits are ignored to align the address. The start address written to the CRC32\_STARTADDRESS register must be the memory address corresponding to the CLA memory map.

---

**NOTE:** If the start address is not within the CLA program ROM space of the CLA memory map, then the calculation will not be initiated.

---

### 7.2.2 Seed

The seed is the initial value used for the CRC-32 calculation. Therefore, the result will vary with the initial seed. The seed can be written into the CRC32\_SEED register.

### 7.2.3 Halt

Once the CRC calculation is triggered, it can be halted by setting the HALT bit in the configuration register (CRC32\_CONTROLREG). When this bit is cleared, the CRC calculation will resume from the current address.

### 7.2.4 Result and Comparison

To protect the contents of the CLA program ROM, which includes TI proprietary code, the result of the CRC-32 must be protected. To protect the result, the actual CRC-32 value as it is calculated cycle by cycle is not made visible. However, the final result of the block is populated in the CRC32\_CRCRESULT register. This is the reason for the block size requirements which must specifically be a minimum of 1KB and an increment of 1KB.

To check the correctness of the CRC-32 result, a golden CRC value should be written in the CRC32\_GOLDENCRC register. After the CRC-32 calculation is completed by the module for the configured block size, the golden CRC will be compared with the final result and the module will then set a pass or fail bit accordingly in the status register (CRC32\_STATUSREG).

## 7.3 CLAPROM Registers

### 7.3.1 CLAPROM Base Addresses

**Table 7-1. CLA PROM CRC Base Address Table**

Device Registers	Register Name	Start Address	End Address
ClaPromCrc32Regs	CLA_PROM_CRC32_REGS	0x000061C0	0x000061DF

### 7.3.2 CLA\_PROM\_CRC32\_REGS Registers

Table 7-2 lists the CLA\_PROM\_CRC32\_REGS registers. All register offset addresses not listed in Table 7-2 should be considered as reserved locations and the register contents should not be modified.

**Table 7-2. CLA\_PROM\_CRC32\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	CRC32_CONTROLREG	CRC32-Control Register	EALLOW	<a href="#">Go</a>
2h	CRC32_STARTADDRESS	CRC32-Start address register	EALLOW	<a href="#">Go</a>
4h	CRC32_SEED	CRC32-Seed Register	EALLOW	<a href="#">Go</a>
6h	CRC32_STATUSREG	CRC32-Status Register		<a href="#">Go</a>
8h	CRC32_CRCRESULT	CRC32-CRC result Register		<a href="#">Go</a>
Ah	CRC32_GOLDENCRC	CRC32-Golden CRC register		<a href="#">Go</a>
18h	CRC32_INTEN	CRC32-Interrupt enable register	EALLOW	<a href="#">Go</a>
1Ah	CRC32_FLG	CRC32-Interrupt Flag Register		<a href="#">Go</a>
1Ch	CRC32_CLR	CRC32-Interrupt Clear Register		<a href="#">Go</a>
1Eh	CRC32_FRC	CRC32-Interrupt Force Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 7-3 shows the codes that are used for access types in this section.

**Table 7-3. CLA\_PROM\_CRC32\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**7.3.2.1 CRC32\_CONTROLREG Register (Offset = 0h) [reset = 0h]**

 CRC32\_CONTROLREG is shown in [Figure 7-2](#) and described in [Table 7-4](#).

 Return to the [Summary Table](#).

CRC32-Control Register

**Figure 7-2. CRC32\_CONTROLREG Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED	BLOCKSIZE						
R-0-0h				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED							HALT
R-0-0h							R/W-0h
7	6	5	4	3	2	1	0
RESERVED			FREE_SOFT	RESERVED			START
R-0-0h			R/W-0h	R-0-0h			R-0/W1S-0h

**Table 7-4. CRC32\_CONTROLREG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R-0	0h	Reserved
22-16	BLOCKSIZE	R/W	0h	Block size: 0x0 : 1 KB (default) 0x1 : 2 KB 0x2 : 3 KB ... 0x7F : 128KB  Note : If the value written to this register is greater than size of ROM then it is internally set to max allowed block size Reset type: CPU1.SYSRSn
15-9	RESERVED	R-0	0h	Reserved
8	HALT	R/W	0h	Halt Bit : 0 : CRC calaculation will resume from where is has hanlted 1 : CRC calculation will be halted  Notes: This bit has effect only after CRC calc is triggered by writing to START Bit Reset type: CPU1.SYSRSn
7-5	RESERVED	R-0	0h	Reserved
4	FREE_SOFT	R/W	0h	emulation control bit : This bit controls behaviour of CRC calculation during emulations 0 : Soft, CRC module stops on MCLA dedug suspend. 1 : Free, CRC calucation is not affected by DEBUG HALT of CLA Reset type: CPU1.SYSRSn
3-1	RESERVED	R-0	0h	Reserved

**Table 7-4. CRC32\_CONTROLREG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	START	R-0/W1S	0h	Start Bit: 0: No effect 1: will start the CRC calculations  Notes: Setting this anytime during the CRC calculation will reset and re-start the CRC calculation. Reset type: CPU1.SYSRSn

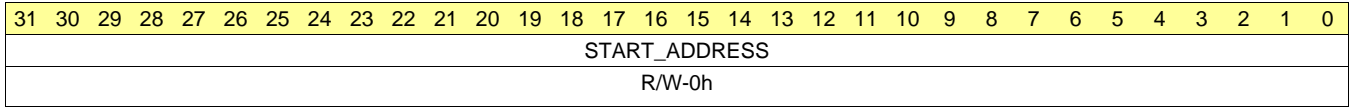
**7.3.2.2 CRC32\_STARTADDRESS Register (Offset = 2h) [reset = 0h]**

CRC32\_STARTADDRESS is shown in [Figure 7-3](#) and described in [Table 7-5](#).

Return to the [Summary Table](#).

CRC32-Start address register

**Figure 7-3. CRC32\_STARTADDRESS Register**



**Table 7-5. CRC32\_STARTADDRESS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	START_ADDRESS	R/W	0h	START_ADDRESS defines starting point for CRC32 calculation. Note that it is CLA address Reset type: CPU1.SYSRSn

### 7.3.2.3 CRC32\_SEED Register (Offset = 4h) [reset = 0h]

CRC32\_SEED is shown in [Figure 7-4](#) and described in [Table 7-6](#).

Return to the [Summary Table](#).

CRC32-Seed Register

**Figure 7-4. CRC32\_SEED 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
SEED																															
R/W-0h																															

**Table 7-6. CRC32\_SEED Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SEED	R/W	0h	Initial value of CRC, this value is copied to the CRC register on triggering CRC calculation by writing to START bit. Reset type: CPU1.SYSRSn



**7.3.2.4 CRC32\_STATUSREG Register (Offset = 6h) [reset = 0h]**

CRC32\_STATUSREG is shown in [Figure 7-5](#) and described in [Table 7-7](#).

Return to the [Summary Table](#).

CRC32-Status Register

**Figure 7-5. CRC32\_STATUSREG Register**

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

**Table 7-7. CRC32\_STATUSREG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RUNSTATUS	R	0h	Status bit: 0 : CRC module is IDLE 1 : CRC module is Active Reset type: CPU1.SYSRSn
30-24	RESERVED	R-0	0h	Reserved
23	CRCCHECKSTATUS	R	0h	CRC pass/fail status bit: 0 : PASS 1 : FAIL Note: Comparion is enabled only after CRC calc is completed Reset type: CPU1.SYSRSn
22-16	RESERVED	R-0	0h	Reserved
15-0	CURRENTADDR	R	0h	The current address of data fetch unit - this is 32 bit aligned offset address of ROM Reset type: CPU1.SYSRSn

### 7.3.2.5 CRC32\_CRCRESULT Register (Offset = 8h) [reset = 0h]

CRC32\_CRCRESULT is shown in [Figure 7-6](#) and described in [Table 7-8](#).

Return to the [Summary Table](#).

CRC32-CRC result Register

**Figure 7-6. CRC32\_CRCRESULT 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
CRCRESULT																															
R-0h																															

**Table 7-8. CRC32\_CRCRESULT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CRCRESULT	R	0h	CRC result register Reset type: CPU1.SYSRSn

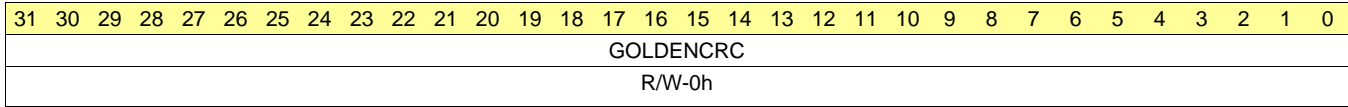
**7.3.2.6 CRC32\_GOLDENCRC Register (Offset = Ah) [reset = 0h]**

CRC32\_GOLDENCRC is shown in [Figure 7-7](#) and described in [Table 7-9](#).

Return to the [Summary Table](#).

CRC32-Golden CRC register

**Figure 7-7. CRC32\_GOLDENCRC Register**



**Table 7-9. CRC32\_GOLDENCRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GOLDENCRC	R/W	0h	Golden CRC register: Value written to this register is compared with CRCRESULT when CRCDONE bit is set.  After the CRC is done, GOLDENCRC will be compared with CRCRESULT and the CRCCHECKSTATUS bit will be updated. Reset type: CPU1.SYSRSn

**7.3.2.7 CRC32\_INTEN Register (Offset = 18h) [reset = 0h]**

CRC32\_INTEN is shown in [Figure 7-8](#) and described in [Table 7-10](#).

Return to the [Summary Table](#).

CRC32-Interrupt enable register

**Figure 7-8. CRC32\_INTEN Register**

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

**Table 7-10. CRC32\_INTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	CRCDONE	R/W	0h	0 CRCDONE Interrupt disabled 1 CRCDONE Interrupt enabled Reset type: CPU1.SYSRSn
0	RESERVED	R-0	0h	Reserved

**7.3.2.8 CRC32\_FLG Register (Offset = 1Ah) [reset = 0h]**

CRC32\_FLG is shown in [Figure 7-9](#) and described in [Table 7-11](#).

Return to the [Summary Table](#).

CRC32-Interrupt Flag Register

**Figure 7-9. CRC32\_FLG Register**

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

**Table 7-11. CRC32\_FLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	CRCDONE	R	0h	Error Interrupt Status flag 0 CRC calculation is in progress or CRC module is idle. 1 CRC calculation is done. Reset type: CPU1.SYSRSn
0	INT	R	0h	Global Interrupt Status flag 0 No interrupt generated 1 Interrupt was generated Reset type: CPU1.SYSRSn

**7.3.2.9 CRC32\_CLR Register (Offset = 1Ch) [reset = 0h]**

CRC32\_CLR is shown in [Figure 7-10](#) and described in [Table 7-12](#).

Return to the [Summary Table](#).

CRC32-Interrupt Clear Register

**Figure 7-10. CRC32\_CLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						CRCDONE	INT
R-0-0h						R-0/W1S-0h	R-0/W1S-0h

**Table 7-12. CRC32\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	CRCDONE	R-0/W1S	0h	Clear CRCDONE interrupt flag 0 No effect 1 Clears the CRCDONE interrupt flag Reset type: CPU1.SYSRSn
0	INT	R-0/W1S	0h	Global Interrupt Clear 0 No effect 1 Clears the interrupt flag and enables further interrupts to be generated if an event flags is set to 1. Reset type: CPU1.SYSRSn

**7.3.2.10 CRC32\_FRC Register (Offset = 1Eh) [reset = 0h]**

CRC32\_FRC is shown in [Figure 7-11](#) and described in [Table 7-13](#).

Return to the [Summary Table](#).

CRC32-Interrupt Force Register

**Figure 7-11. CRC32\_FRC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						CRCDONE	RESERVED
R-0-0h						R-0/W1S-0h	R-0-0h

**Table 7-13. CRC32\_FRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	CRCDONE	R-0/W1S	0h	Force CRCDONE interrupt flag 0 No effect 1 Force CRCDONE interrupt flag Reset type: CPU1.SYSRSn
0	RESERVED	R-0	0h	Reserved

## General-Purpose Input/Output (GPIO)

---



---

The GPIO module controls the device's digital and analog I/O multiplexing, which uses shared pins to maximize application flexibility. The pins are named by their general-purpose I/O name (for example, GPIO0, GPIO25, GPIO58). These pins can be individually selected to operate as digital I/O (also called GPIO mode), or connected to one of several peripheral I/O signals. The input signals can be qualified to remove unwanted noise.

Topic	Page
<b>8.1 Introduction .....</b>	<b>881</b>
<b>8.2 Configuration Overview .....</b>	<b>882</b>
<b>8.3 Digital General-Purpose I/O Control .....</b>	<b>883</b>
<b>8.4 Input Qualification .....</b>	<b>884</b>
<b>8.5 GPIO and Peripheral Muxing .....</b>	<b>887</b>
<b>8.6 Internal Pullup Configuration Requirements .....</b>	<b>893</b>
<b>8.7 GPIO Registers.....</b>	<b>894</b>



### 8.1 Introduction

Up to twelve independent peripheral signals are multiplexed on a single GPIO-enabled pin in addition to the CPU-controlled I/O capability. Each pin output can be controlled by either a peripheral or one of the two CPU masters (CPU1, CPU1.CLA). There are two I/O ports:

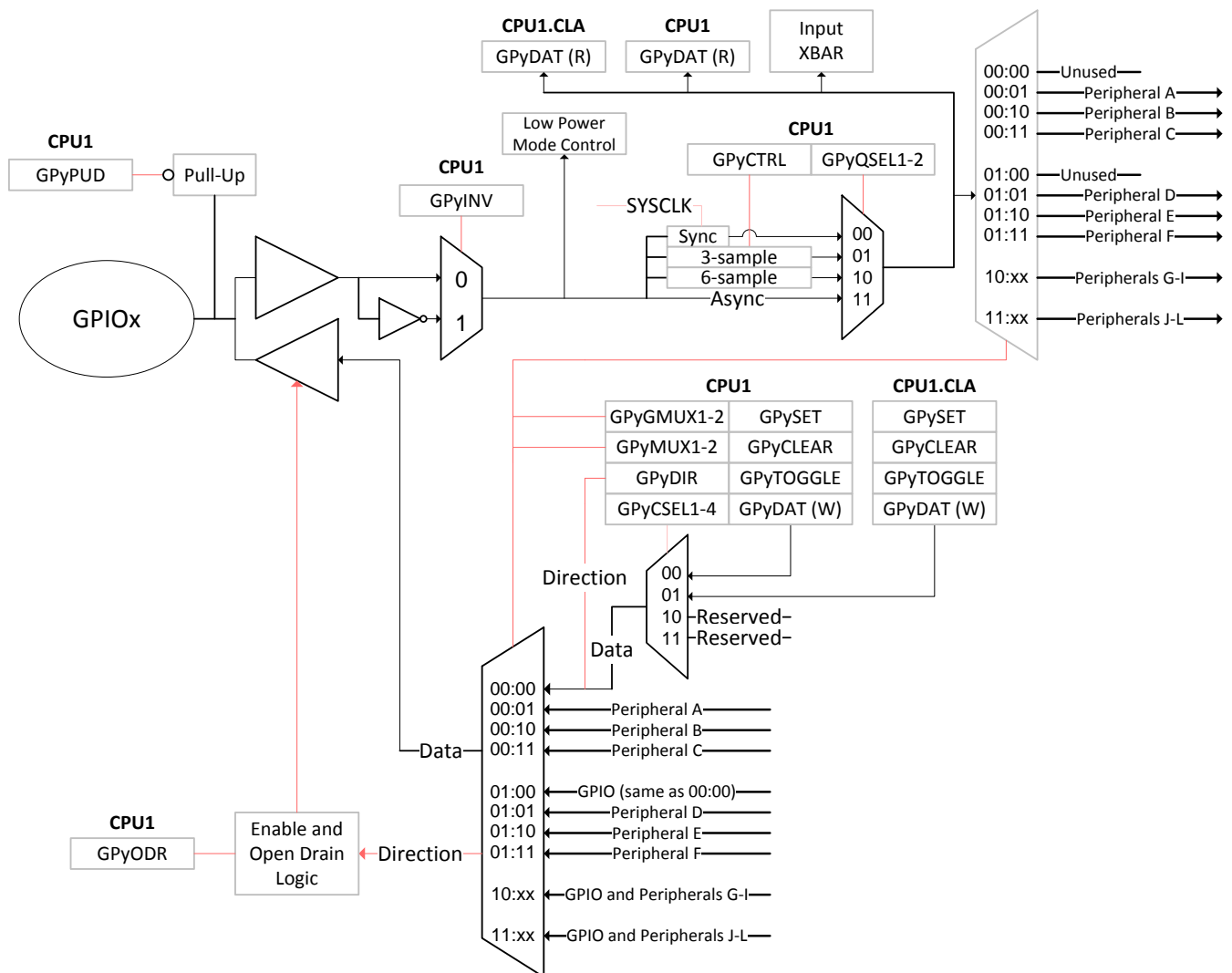
- Port A consists of GPIO0-GPIO31
- Port B consists of GPIO32-GPIO59

The analog signals on this device are multiplexed with digital inputs. These analog IO (AIO) pins do not have digital output capability. They are assigned to a single port:

- Port H consists of GPIO224-GPIO247

Figure 8-1 shows the GPIO logic for a single pin.

Figure 8-1. GPIO Logic for a Single Pin



**NOTE:** High-speed SPI and AUXCLKIN use a different signal path that does not support inversion or qualification. For more details on high-speed SPI pins, see .

The USB PHY pin muxing is not shown in this diagram. For more details on USB pins, see .

There are two key features to note in this diagram. The first is that the input and output paths are entirely separate, connecting only at the pin. The second is that peripheral muxing takes place far from the pin. As a result, it is always possible for both CPUs and CLAs to read the physical state of the pin independent of CPU mastering and peripheral muxing. Likewise, external interrupts can be generated from peripheral activity. All pin options such as input qualification and open-drain output are valid for all masters and peripherals. However, the peripheral muxing, CPU muxing, and pin options can only be configured by CPU1.

A separate configuration is required for the USB signals. See for details.

---

**NOTE:** JTAG uses a different signal path that does not support inversion or qualification.

---

**NOTE:** GPIO22 and GPIO23 are in a special analog mode at reset, and must be reconfigured for GPIO use by disabling DC-DC and clearing their bits in GPAAMSEL. GPIO23's maximum toggle frequency is limited. The exact limit is TBD, but estimated to be 12 MHz.

---

**NOTE:** GPIO18/X2 has different timings due to the load placed on it by the oscillator circuit. For information on using GPIO18/X2 as a GPIO, please see the device datasheet and the Clocking section of this document.

---

**NOTE:** If digital signals with sharp edges (high dv/dt) are connected to the AIOs, cross-talk can occur with adjacent analog signals. Therefore, the user should limit the edge rate of signals connected to AIOs if adjacent channels are being used for analog functions.

---

## 8.2 Configuration Overview

I/O pin configuration consists of several steps:

### 1. Plan the device pin-out

Make a list of all required peripherals for the application. Using the peripheral mux information in the device data manual, choose which GPIOs to use for the peripheral signals. Decide which of the remaining GPIOs to use as inputs and outputs for each CPU and CLA.

Once the peripheral muxing has been chosen, it should be implemented by writing the appropriate values to the GPyMUX1/2 and GPyGMUX1/2 registers. When changing the GPyGMUX value for a pin, always set the corresponding GPyMUX bits to zero first to avoid glitching in the muxes. By default, all pins are general-purpose I/Os, not peripheral signals.

### 2. (Optional) Enable internal pullup resistors

To enable or disable the pullup resistors, write to the appropriate bits in the GPIO pullup disable registers (GPyPUD). All pullups are disabled by default. pullups can be used to keep input pins in a known state when there is no external signal driving them.

### 3. Select input qualification

If the pin will be used as an input, specify the required input qualification, if any. The input qualification sampling period is selected in the GPyCTRL registers, while the type of qualification is selected in the GPyQSEL1 and GPyQSEL2 registers. By default, all qualification is synchronous with a sampling period equal to PLLSYSCLK. For an explanation of input qualification, see section [Section 8.4](#).

### 4. Select the direction of any general-purpose I/O pins

For each pin configured as a GPIO, specify the direction of the pin as either input or output using the GPyDIR registers. By default, all GPIO pins are inputs. Before changing a pin to an output, load the output latch with the value to be driven by writing that value to the GPySET, GPyCLEAR, or GPyDAT registers. Once the latch is loaded, write to GPyDIR to change the pin direction. By default, all output latches are zero.

### 5. Select low-power mode wake-up sources

GPIOs 0-59 can be used to wake the system up from Standby or Halt mode. To select one or more GPIOs for wake-up, write to the appropriate bits in the GPIOLPMSEL0 and GPIOLPMSEL1 registers. These registers are part of the CPU system register space. For more information on low-power modes

and GPIO wake-up, see the Low Power Modes section in the *System Control and Interrupts* chapter.

#### 6. Select external interrupt sources

Configuring external interrupts is a two-step process. First, the interrupts themselves must be enabled and their polarity must be configured via the XINTnCR registers. Second, the XINT1-5 GPIO pins must be set by selecting the sources for Input X-BAR signals 4, 5, 6, 13, and 14, respectively. For more information on the Input X-BAR architecture, see the *Crossbar(XBAR)* chapter of this manual.

### 8.3 Digital General-Purpose I/O Control

The values on the pins that are configured as GPIO can be changed by using these registers.

- **GPyDAT Registers**

Each I/O port has one data register. Each bit in the data register corresponds to one GPIO pin. No matter how the pin is configured (GPIO or peripheral function), the corresponding bit in the data register reflects the current state of the pin after qualification. Writing to the GPyDAT register clears or sets the corresponding output latch and if the pin is enabled as a general purpose output (GPIO output) the pin will also be driven either low or high. If the pin is not configured as a GPIO output then the value will be latched, but the pin will not be driven. Only if the pin is later configured as a GPIO output, will the latched value be driven onto the pin.

When using the GPyDAT register to change the level of an output pin, you should be cautious not to accidentally change the level of another pin. For example, if you mean to change the output latch level of GPIOA1 by writing to the GPADAT register bit 0 using a read-modify-write instruction, a problem can occur if another I/O port A signal changes level between the read and the write stage of the instruction. Following is an analysis of why this happens:

The GPyDAT registers reflect the state of the pin, not the latch. This means the register reflects the actual pin value. However, there is a lag between when the register is written to when the new pin value is reflected back in the register. This may pose a problem when this register is used in subsequent program statements to alter the state of GPIO pins. An example is shown below where two program statements attempt to drive two different GPIO pins that are currently low to a high state.

If Read-Modify-Write operations are used on the GPyDAT registers, because of the delay between the output and the input of the first instruction (I1), the second instruction (I2) will read the old value and write it back.

```
GpioDataRegs.GPADAT.bit.GPIO1 = 1; //I1 performs read-modify-
write of GPADAT GpioDataRegs.GPADAT.bit.GPIO2 = 1; //I2 also a read-modify-
write of GPADAT. //GPADAT gets the old value of GPIO1 due to the delay
```

The second instruction will wait for the first to finish its write due to the write-followed-by-read protection on this peripheral frame. There will be some lag, however, between the write of (I1) and the GPyDAT bit reflecting the new value (1) on the pin. During this lag, the second instruction will read the old value of GPIO1 (0) and write it back along with the new value of GPIO2 (1). Therefore, GPIO1 pin stays low.

One solution is to put some NOPs between instructions. A better solution is to use the GPySET/GPyCLEAR/GPyTOGGLE registers instead of the GPyDAT registers. These registers always read back a 0 and writes of 0 have no effect. Only bits that need to be changed can be specified without disturbing any other bit(s) that are currently in the process of changing.

- **GPySET Registers**

The set registers are used to drive specified GPIO pins high without disturbing other pins. Each I/O port has one set register and each bit corresponds to one GPIO pin. The set registers always read back 0. If the corresponding pin is configured as an output, then writing a 1 to that bit in the set register will set the output latch high and the corresponding pin will be driven high. If the pin is not configured as a GPIO output, then the value will be latched but the pin will not be driven. Only if the pin is later configured as a GPIO output will the latched value will be driven onto the pin. Writing a 0 to any bit in the set registers has no effect.

- **GPyCLEAR Registers**

The clear registers are used to drive specified GPIO pins low without disturbing other pins. Each I/O port has one clear register. The clear registers always read back 0. If the corresponding pin is configured as a general purpose output, then writing a 1 to the corresponding bit in the clear register will clear the output latch and the pin will be driven low. If the pin is not configured as a GPIO output,

then the value will be latched but the pin will not be driven. Only if the pin is later configured as a GPIO output will the latched value will be driven onto the pin. Writing a 0 to any bit in the clear registers has no effect.

- **GPyTOGGLE Registers**

The toggle registers are used to drive specified GPIO pins to the opposite level without disturbing other pins. Each I/O port has one toggle register. The toggle registers always read back 0. If the corresponding pin is configured as an output, then writing a 1 to that bit in the toggle register flips the output latch and pulls the corresponding pin in the opposite direction. That is, if the output pin is driven low, then writing a 1 to the corresponding bit in the toggle register will pull the pin high. Likewise, if the output pin is high, then writing a 1 to the corresponding bit in the toggle register will pull the pin low. If the pin is not configured as a GPIO output, then the value will be latched but the pin will not be driven. Only if the pin is later configured as a GPIO output will the latched value will be driven onto the pin. Writing a 0 to any bit in the toggle registers has no effect.

## 8.4 Input Qualification

The input qualification scheme has been designed to be very flexible. You can select the type of input qualification for each GPIO pin by configuring the GPyQSEL1 and GPyQSEL2 registers. In the case of a GPIO input pin, the qualification can be specified as only synchronize to SYSCLKOUT or qualification by a sampling window. For pins that are configured as peripheral inputs, the input can also be asynchronous in addition to synchronized to SYSCLKOUT or qualified by a sampling window. The remainder of this section describes the options available.

### 8.4.1 No Synchronization (Asynchronous Input)

This mode is used for peripherals where input synchronization is not required or the peripheral itself performs the synchronization. Examples include communication ports McBSP, SCI, SPI, and I<sup>2</sup>C. In addition, it may be desirable to have the ePWM trip zone ( $\overline{TZn}$ ) signals function independent of the presence of SYSCLKOUT.

The asynchronous option is not valid if the pin is used as a general purpose digital input pin (GPIO). If the pin is configured as a GPIO input and the asynchronous option is selected then the qualification defaults to synchronization to SYSCLKOUT as described in [Section 8.4.2](#).

---

**NOTE:** Using input synchronization when the peripheral itself performs the synchronization may cause unexpected results. The user should ensure that the GPIO pin is configured for asynchronous in this case.

---

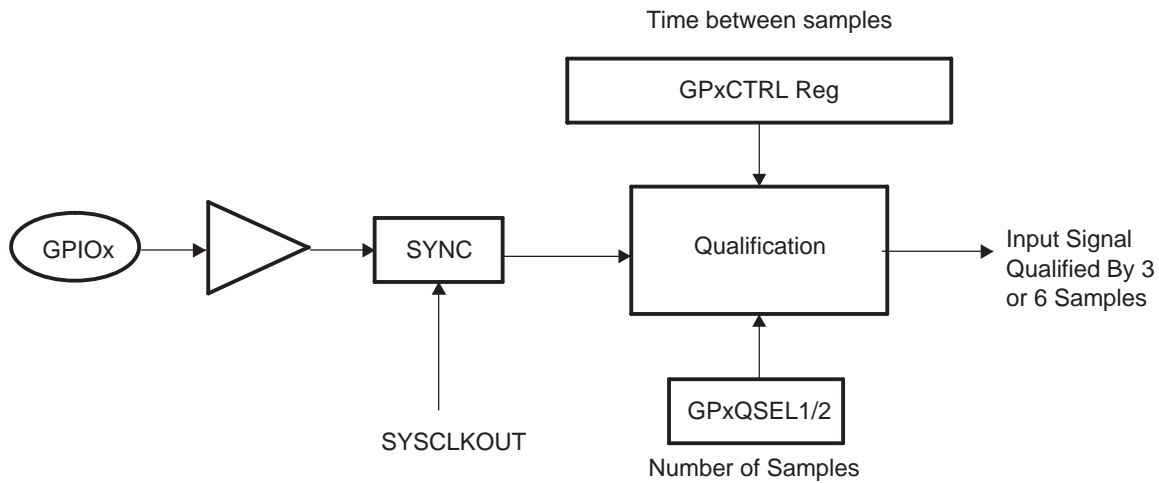
### 8.4.2 Synchronization to SYSCLKOUT Only

This is the default qualification mode of all the pins at reset. In this mode, the input signal is only synchronized to the system clock (SYSCLKOUT). Because the incoming signal is asynchronous, it can take up to a SYSCLKOUT period of delay in order for the input to the MCU to be changed. No further qualification is performed on the signal.

### 8.4.3 Qualification Using a Sampling Window

In this mode, the signal is first synchronized to the system clock (SYSCLKOUT) and then qualified by a specified number of cycles before the input is allowed to change. [Figure 8-2](#) and [Figure 8-3](#) show how the input qualification is performed to eliminate unwanted noise. Two parameters are specified by the user for this type of qualification: 1) the sampling period, or how often the signal is sampled, and 2) the number of samples to be taken.

**Figure 8-2. Input Qualification Using a Sampling Window**



**Time between samples (sampling period):**

To qualify the signal, the input signal is sampled at a regular period. The sampling period is specified by the user and determines the time duration between samples, or how often the signal will be sampled, relative to the CPU clock (SYSCLKOUT).

The sampling period is specified by the qualification period (QUALPRDn) bits in the GPxCTRL register. The sampling period is configurable in groups of 8 input signals. For example, GPIO0 to GPIO7 use GPCCTRL[QUALPRD0] setting and GPIO8 to GPIO15 use GPCCTRL[QUALPRD1]. [Table 8-1](#) and [Table 8-2](#) show the relationship between the sampling period or sampling frequency and the GPxCTRL[QUALPRDn] setting.

**Table 8-1. Sampling Period**

	Sampling Period
If GPxCTRL[QUALPRDn] = 0	$1 \times T_{\text{SYSCLKOUT}}$
If GPxCTRL[QUALPRDn] ≠ 0	$2 \times \text{GPxCTRL[QUALPRDn]} \times T_{\text{SYSCLKOUT}}$
	Where $T_{\text{SYSCLKOUT}}$ is the period in time of SYSCLKOUT

**Table 8-2. Sampling Frequency**

	Sampling Frequency
If GPxCTRL[QUALPRDn] = 0	$f_{\text{SYSCLKOUT}}$
If GPxCTRL[QUALPRDn] ≠ 0	$f_{\text{SYSCLKOUT}} \times 1 \div (2 \times \text{GPxCTRL[QUALPRDn]})$
	Where $f_{\text{SYSCLKOUT}}$ is the frequency of SYSCLKOUT

From these equations, the minimum and maximum time between samples can be calculated for a given SYSCLKOUT frequency:

**Example: Maximum Sampling Frequency:**

If GPxCTRL[QUALPRDn] = 0  
then the sampling frequency is  $f_{\text{SYSCLKOUT}}$   
If, for example,  $f_{\text{SYSCLKOUT}} = 60 \text{ MHz}$   
then the signal will be sampled at 60 MHz or one sample every 16.67 ns.

**Example: Minimum Sampling Frequency:**

If GPxCTRL[QUALPRDn] = 0xFF (255)

then the sampling frequency is  $f_{\text{SYSCLKOUT}} \times 1 \div (2 \times \text{GPxCTRL[QUALPRDn]})$

If, for example,  $f_{\text{SYSCLKOUT}} = 60 \text{ MHz}$

then the signal will be sampled at  $60 \text{ MHz} \times 1 \div (2 \times 255)$  or one sample every 8.5  $\mu\text{s}$ .

**Number of samples:**

The number of times the signal is sampled is either three samples or six samples as specified in the qualification selection (GPAQSEL1, GPAQSEL2, GPBQSEL1, and GPBQSEL2) registers. When three or six consecutive cycles are the same, then the input change will be passed through to the MCU.

**Total Sampling Window Width:**

The sampling window is the time during which the input signal will be sampled as shown in [Figure 8-3](#). By using the equation for the sampling period along with the number of samples to be taken, the total width of the window can be determined.

For the input qualifier to detect a change in the input, the level of the signal must be stable for the duration of the sampling window width or longer.

The number of sampling periods within the window is always one less than the number of samples taken. For a three-sample window, the sampling window width is two sampling periods wide where the sampling period is defined in [Table 8-1](#). Likewise, for a six-sample window, the sampling window width is five sampling periods wide. [Table 8-3](#) and [Table 8-4](#) show the calculations that can be used to determine the total sampling window width based on GPxCTRL[QUALPRDn] and the number of samples taken.

**Table 8-3. Case 1: Three-Sample Sampling Window Width**

	Total Sampling Window Width
If GPxCTRL[QUALPRDn] = 0	$2 \times T_{\text{SYSCLKOUT}}$
If GPxCTRL[QUALPRDn] $\neq$ 0	$2 \times 2 \times \text{GPxCTRL[QUALPRDn]} \times T_{\text{SYSCLKOUT}}$
	Where $T_{\text{SYSCLKOUT}}$ is the period in time of SYSCLKOUT

**Table 8-4. Case 2: Six-Sample Sampling Window Width**

	Total Sampling Window Width
If GPxCTRL[QUALPRDn] = 0	$5 \times T_{\text{SYSCLKOUT}}$
If GPxCTRL[QUALPRDn] $\neq$ 0	$5 \times 2 \times \text{GPxCTRL[QUALPRDn]} \times T_{\text{SYSCLKOUT}}$
	Where $T_{\text{SYSCLKOUT}}$ is the period in time of SYSCLKOUT

**NOTE:** The external signal change is asynchronous with respect to both the sampling period and SYSCLKOUT. Due to the asynchronous nature of the external signal, the input should be held stable for a time greater than the sampling window width to make sure the logic detects a change in the signal. The extra time required can be up to an additional sampling period +  $T_{\text{SYSCLKOUT}}$ .

The required duration for an input signal to be stable for the qualification logic to detect a change is described in the device-specific data manual.

**Example Qualification Window:**

For the example shown in [Figure 8-3](#), the input qualification has been configured as follows:

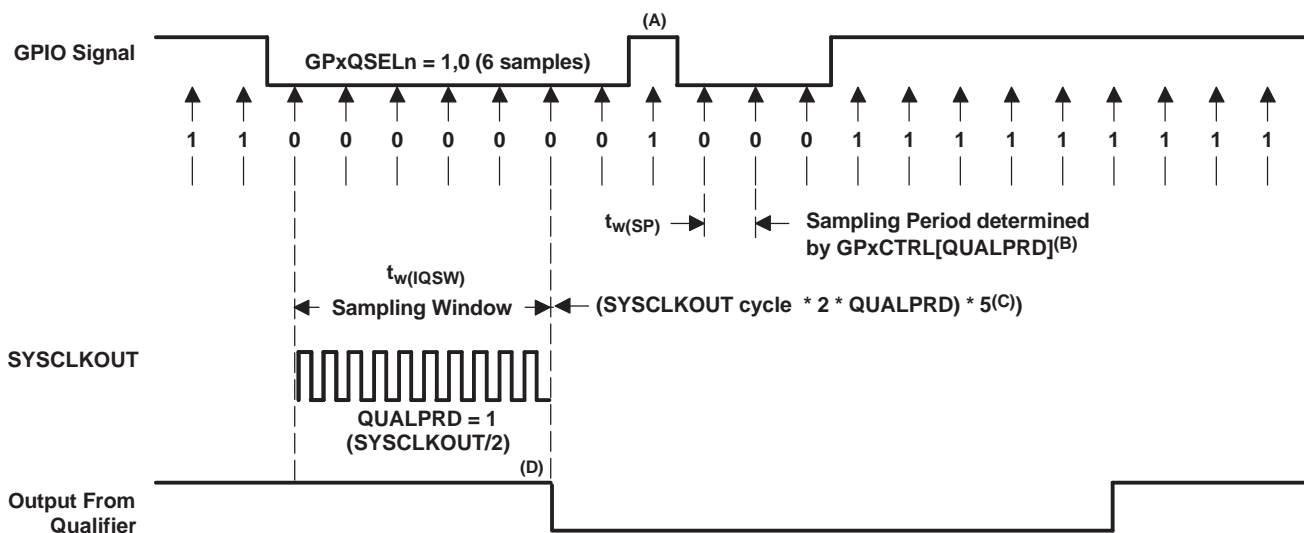
- GPxQSEL1/2 = 1,0. This indicates a six-sample qualification.
- GPxCTRL[QUALPRDn] = 1. The sampling period is  $t_w(\text{SP}) = 2 \times \text{GPxCTRL[QUALPRDn]} \times T_{\text{SYSCLKOUT}} = 2 \times T_{\text{SYSCLKOUT}}$ .

This configuration results in the following:



- The width of the sampling window is: .  
 $t_w(IQSW) = 5 \times t_w(SP) = 5 \times 2 \times GPxCTRL[QUALPRDn] \times T_{SYCLKOUT} = 5 \times 2 \times T_{SYCLKOUT}$
- If, for example,  $T_{SYCLKOUT} = 16.67 \text{ ns}$ , then the duration of the sampling window is:  
 Sampling period,  $t_w(SP) = 2 \times T_{SYCLKOUT} = 2 \times 16.67 \text{ ns} = 33.3 \text{ ns}$   
 Sampling window,  $t_w(IQSW) = 5 \times t_w(SP) = 5 \times 33.3 \text{ ns} = 166.7 \text{ ns}$ .
- To account for the asynchronous nature of the input relative to the sampling period and SYCLKOUT, up to an additional sampling period and SYCLK period may be required to detect a change in the input signal. For this example:  
 $t_w(IQSW) + t_w(SP) + T_{SYCLKOUT} = 166.7 \text{ ns} + 33.3 \text{ ns} + 16.67 \text{ ns} = 216.7 \text{ ns}$
- In Figure 8-3, the glitch (A) is shorter than the qualification window and will be ignored by the input qualifier.

Figure 8-3. Input Qualifier Clock Cycles



- This glitch will be ignored by the input qualifier. The QUALPRD bit field specifies the qualification sampling period. It can vary from 00 to 0xFF. If QUALPRD = 00, then the sampling period is 1 SYCLKOUT cycle. For any other value "n", the qualification sampling period is 2n SYCLKOUT cycles (i.e., at every 2n SYCLKOUT cycles, the GPIO pin will be sampled).
- The qualification period selected via the GPxCTRL register applies to groups of 8 GPIO pins.
- The qualification block can take either three or six samples. The GPxQSELn Register selects which sample mode is used.
- In the example shown, for the qualifier to detect the change, the input should be stable for 10 SYCLKOUT cycles or greater. In other words, the inputs should be stable for (5 x QUALPRD x 2) SYCLKOUT cycles. That would ensure 5 sampling periods for detection to occur. Since external signals are driven asynchronously, a 13-SYCLKOUT-wide pulse ensures reliable recognition.

## 8.5 GPIO and Peripheral Muxing

Up to twelve different peripheral functions are multiplexed to each pin along with a general-purpose input/output (GPIO) function. This allows you to choose the peripheral mix and pinout that will work best for your particular application. Refer to the table below for muxing combinations and definitions.

**Table 8-5. GPIO Muxed Pins**

0, 4, 8, 12	1	2	3	5	6	7	9	10	11	13	14	15
GPIO0	EPWM1_A				I2CA_SDA							
GPIO1	EPWM1_B				I2CA_SCL							
GPIO2	EPWM2_A			OUTPUTXBAR1	PMBUSA_SDA		SCIA_TX	FSIRXA_D1				
GPIO3	EPWM2_B	OUTPUTXBAR2		OUTPUTXBAR2	PMBUSA_SCL	SPIA_CLK	SCIA_RX	FSIRXA_D0				
GPIO4	EPWM3_A			OUTPUTXBAR3	CANA_TX			FSIRXA_CLK				
GPIO5	EPWM3_B		OUTPUTXBAR3		CANA_RX	SPIA_STE	FSITXA_D1					
GPIO6	EPWM4_A	OUTPUTXBAR4	SYNCOUT	EQEP1_A	CANB_TX	SPIB_SOMI	FSITXA_D0					
GPIO7	EPWM4_B		OUTPUTXBAR5	EQEP1_B	CANB_RX	SPIB_SIMO	FSITXA_CLK					
GPIO8	EPWM5_A	CANB_TX	ADCSOCAO	EQEP1_STROBE	SCIA_TX	SPIA_SIMO	I2CA_SCL	FSITXA_D1				
GPIO9	EPWM5_B	SCIB_TX	OUTPUTXBAR6	EQEP1_INDEX	SCIA_RX	SPIA_CLK		FSITXA_D0				
GPIO10	EPWM6_A	CANB_RX	ADCSOCBO	EQEP1_A	SCIB_TX	SPIA_SOMI	I2CA_SDA	FSITXA_CLK				
GPIO11	EPWM6_B	SCIB_RX	OUTPUTXBAR7	EQEP1_B	SCIB_RX	SPIA_STE	FSIRXA_D1					
GPIO12	EPWM7_A	CANB_TX		EQEP1_STROBE	SCIB_TX	PMBUSA_CTL	FSIRXA_D0					
GPIO13	EPWM7_B	CANB_RX		EQEP1_INDEX	SCIB_RX	PMBUSA_ALERT	FSIRXA_CLK					
GPIO14	EPWM8_A	SCIB_TX			OUTPUTXBAR3	PMBUSA_SDA	SPIB_CLK	EQEP2_A				
GPIO15	EPWM8_B	SCIB_RX			OUTPUTXBAR4	PMBUSA_SCL	SPIB_STE	EQEP2_B				
GPIO16	SPIA_SIMO	CANB_TX	OUTPUTXBAR7	EPWM5_A	SCIA_TX	SD1_D1	EQEP1_STROBE	PMBUSA_SCL	XCLKOUT			
GPIO17	SPIA_SOMI	CANB_RX	OUTPUTXBAR8	EPWM5_B	SCIA_RX	SD1_C1	EQEP1_INDEX	PMBUSA_SDA				
GPIO18_X2	SPIA_CLK	SCIB_TX	CANA_RX	EPWM6_A	I2CA_SCL	SD1_D2	EQEP2_A	PMBUSA_CTL	XCLKOUT			
GPIO20												
GPIO21												
GPIO22_VFBSW	EQEP1_STROBE		SCIB_TX			SPIB_CLK	SD1_D4	LINA_TX				
GPIO23_VSW												
GPIO24	OUTPUTXBAR1	EQEP2_A		EPWM8_A	SPIB_SIMO	SD1_D1		PMBUSA_SCL	SCIA_TX	ERRORSTS		
GPIO25	OUTPUTXBAR2	EQEP2_B			SPIB_SOMI	SD1_C1	FSITXA_D1	PMBUSA_SDA	SCIA_RX			
GPIO26	OUTPUTXBAR3	EQEP2_INDEX		OUTPUTXBAR3	SPIB_CLK	SD1_D2	FSITXA_D0	PMBUSA_CTL	I2CA_SDA			
GPIO27	OUTPUTXBAR4	EQEP2_STROBE		OUTPUTXBAR4	SPIB_STE	SD1_C2	FSITXA_CLK	PMBUSA_ALERT	I2CA_SCL			
GPIO28	SCIA_RX		EPWM7_A	OUTPUTXBAR5	EQEP1_A	SD1_D3	EQEP2_STROBE	LINA_TX	SPIB_CLK	ERRORSTS		
GPIO29	SCIA_TX		EPWM7_B	OUTPUTXBAR6	EQEP1_B	SD1_C3	EQEP2_INDEX	LINA_RX	SPIB_STE	ERRORSTS		
GPIO30	CANA_RX		SPIB_SIMO	OUTPUTXBAR7	EQEP1_STROBE	SD1_D4						
GPIO31	CANA_TX		SPIB_SOMI	OUTPUTXBAR8	EQEP1_INDEX	SD1_C4	FSIRXA_D1					
GPIO32	I2CA_SDA		SPIB_CLK	EPWM8_B	LINA_TX	SD1_D3	FSIRXA_D0	CANA_TX				
GPIO33	I2CA_SCL		SPIB_STE	OUTPUTXBAR4	LINA_RX	SD1_C3	FSIRXA_CLK	CANA_RX				
GPIO34	OUTPUTXBAR1				PMBUSA_SDA							
GPIO35	SCIA_RX		I2CA_SDA	CANA_RX	PMBUSA_SCL	LINA_RX	EQEP1_A	PMBUSA_CTL				TDI



**Table 8-5. GPIO Muxed Pins (continued)**

0, 4, 8, 12	1	2	3	5	6	7	9	10	11	13	14	15
GPIO37	OUTPUTXBAR2		I2CA_SCL	SCIA_TX	CANA_TX	LINA_TX	EQEP1_B	PMBUSA_ALER T				TDO
GPIO39					CANB_RX	FSIRXA_CLK						
GPIO40					PMBUSA_SDA	FSIRXA_D0	SCIB_TX	EQEP1_A				
GPIO41												
GPIO42												
GPIO43												
GPIO44												
GPIO45												
GPIO46												
GPIO47												
GPIO48												
GPIO49												
GPIO50												
GPIO51												
GPIO52												
GPIO53												
GPIO54												
GPIO55												
GPIO56	SPIA_CLK			EQEP2_STROB E	SCIB_TX	SD1_D3	SPIB_SIMO		EQEP1_A			
GPIO57	SPIA_STE			EQEP2_INDEX	SCIB_RX	SD1_C3	SPIB_SOMI		EQEP1_B			
GPIO58				OUTPUTXBAR1	SPIB_CLK	SD1_D4	LINA_TX	CANB_TX	EQEP1_STROB E			
GPIO59				OUTPUTXBAR2	SPIB_STE	SD1_C4	LINA_RX	CANB_RX	EQEP1_INDEX			

For example, the multiplexing for the GPIO 6 pin is controlled by writing to GPAGMUX[13:12] and GPAMUX[13:12]. By writing to these bits, GPIO 6 can be configured as either a general-purpose digital I/O or one of four different peripheral functions. The options are shown in [Table 8-6](#).

**Table 8-6. GPIO and Peripheral Muxing**

<b>GPAGMUX1[13:12]</b>	<b>GPAMUX1[13:12]</b>	<b>Pin functionality</b>
00	00	GPIO6
00	01	EPWM4A
00	10	OUTPUTXBAR4
00	11	EXTSYNCOUT
01	00	GPIO6
01	01	EQEP1A
01	10	CANB_TX
01	11	SPIBSOMI
10	00	GPIO6
10	01	FSITXAD0
10	10	
10	11	
11	00	
11	01	
11	10	
11	11	

The devices have different multiplexing schemes. If a peripheral is not available on a particular device, that mux selection is reserved on that device and should not be used.

---

**NOTE:** If you select a reserved GPIO mux configuration that is not mapped to either a peripheral or GPIO mode, the state of the pin will be undefined and the pin may be driven. Unimplemented configurations are for future expansion and must not be selected. In the device mux table (see datasheet), these options are indicated as Reserved or left blank.

---

Some peripherals can be assigned to more than one pin via the mux registers. For example, OUTPUTXBAR1 can be assigned to GPIOs 2, 24, 34, or 58, depending on individual system requirements. An example of this is shown in [Table 8-7](#).

**Table 8-7. Peripheral Muxing (multiple pins assigned)**

GMUX Configuration	MUX Configuration	
Choice 1 GPIO2	GPAGMUX1[5:4]=01	GPAMUX1[5:4]=01
or Choice 2 GPIO24	GPAGMUX2[17:16]=00	GPAMUX2[17:16]=01
or Choice 3 GPIO34	GPBGMUX1[5:4]=00	GPBMUX1[5:4]=01
or Choice 4 GPIO58	GPBGMUX2[21:20]=01	GPBMUX2[21:20]=01

If none, or more than one, of the GPIO pins are configured as peripheral input pins, then that GPIO will be set to a hard-wired default value.

## 8.6 Internal Pullup Configuration Requirements

On reset, GPIOs are in input mode and have the internal pullups disabled. An un-driven input can float to a mid-rail voltage and cause wasted shoot-through current on the input buffer. The user should always put each GPIO in one of these configurations:

- Input mode and driven on the board by another component to a level above  $V_{ih}$  or below  $V_{il}$
- Input mode with GPIO internal pullup enabled
- Output mode

On devices with lesser pin count packages, pull-ups on unbonded GPIOs are by default enabled to prevent floating inputs. The user should take care to avoid disabling these pullups in their application code.

On devices in the 176 PTP packages, the pullups for any internally unbonded GPIO must be enabled to prevent floating inputs. TI has provided functions in controlSUITE which users can call to enable the pullup on any unbonded GPIO for the package they are using. This function, `GPIO_EnabledUnbondedIOPullups()`, resides in the `(Device)_Sysctrl.c` file and is called by default from `InitSysCtrl()`. The user should take care to avoid disabling these pullups in their application code.

## 8.7 GPIO Registers

This section describes the General Purpose Input/Output Registers.

### 8.7.1 GPIO Base Addresses

**Table 8-8. GPIO Base Address Table**

Device Registers	Register Name	Start Address	End Address
GpioCtrlRegs	GPIO_CTRL_REGS	0x0000_7C00	0x0000_7EFF
GpioDataRegs	GPIO_DATA_REGS	0x0000_7F00	0x0000_7FFF

## 8.7.2 GPIO\_CTRL\_REGS Registers

Table 8-9 lists the GPIO\_CTRL\_REGS registers. All register offset addresses not listed in Table 8-9 should be considered as reserved locations and the register contents should not be modified.

**Table 8-9. GPIO\_CTRL\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	GPACTRL	GPIO A Qualification Sampling Period (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
2h	GPAQSEL1	GPIO A Qualification Type (GPIO0 to GPIO15)	EALLOW	<a href="#">Go</a>
4h	GPAQSEL2	GPIO A Qualification Type (GPIO16 to GPIO31)	EALLOW	<a href="#">Go</a>
6h	GPAMUX1	GPIO A Peripheral Mux (GPIO0 to GPIO15)	EALLOW	<a href="#">Go</a>
8h	GPAMUX2	GPIO A Peripheral Mux (GPIO16 to GPIO31)	EALLOW	<a href="#">Go</a>
Ah	GPADIR	GPIO A Direction (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
Ch	GPAPUD	GPIO A Pull-Up Disable (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
10h	GPAINV	GPIO A Input Inversion (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
12h	GPAODR	GPIO A Open Drain Output Mode (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
14h	GPAAMSEL	GPIO A Analog Mode Select (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
20h	GPAGMUX1	GPIO A Peripheral Group Mux (GPIO0 to GPIO15)	EALLOW	<a href="#">Go</a>
22h	GPAGMUX2	GPIO A Peripheral Group Mux (GPIO16 to GPIO31)	EALLOW	<a href="#">Go</a>
28h	GPACSEL1	GPIO A Master Core Select (GPIO0 to GPIO7)	EALLOW	<a href="#">Go</a>
2Ah	GPACSEL2	GPIO A Master Core Select (GPIO8 to GPIO15)	EALLOW	<a href="#">Go</a>
2Ch	GPACSEL3	GPIO A Master Core Select (GPIO16 to GPIO23)	EALLOW	<a href="#">Go</a>
2Eh	GPACSEL4	GPIO A Master Core Select (GPIO24 to GPIO31)	EALLOW	<a href="#">Go</a>
3Ch	GPALOCK	GPIO A Lock Register (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
3Eh	GPACR	GPIO A Lock Commit Register (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
40h	GPBCTRL	GPIO B Qualification Sampling Period (GPIO32 to GPIO63)	EALLOW	<a href="#">Go</a>
42h	GPBQSEL1	GPIO B Qualification Type (GPIO32 to GPIO47)	EALLOW	<a href="#">Go</a>
44h	GPBQSEL2	GPIO B Qualification Type (GPIO48 to GPIO63)	EALLOW	<a href="#">Go</a>
46h	GPBMUX1	GPIO B Peripheral Mux (GPIO32 to GPIO47)	EALLOW	<a href="#">Go</a>
48h	GPBMUX2	GPIO B Peripheral Mux (GPIO48 to GPIO63)	EALLOW	<a href="#">Go</a>
4Ah	GPBDIR	GPIO B Direction (GPIO32 to GPIO63)	EALLOW	<a href="#">Go</a>
4Ch	GPBPUD	GPIO B Pull-Up Disable (GPIO32 to GPIO63)	EALLOW	<a href="#">Go</a>
50h	GPBINV	GPIO B Input Inversion (GPIO32 to GPIO63)	EALLOW	<a href="#">Go</a>
52h	GPBODR	GPIO B Open Drain Output Mode (GPIO32 to GPIO63)	EALLOW	<a href="#">Go</a>
60h	GPBGMUX1	GPIO B Peripheral Group Mux (GPIO32 to GPIO47)	EALLOW	<a href="#">Go</a>
62h	GPBGMUX2	GPIO B Peripheral Group Mux (GPIO48 to GPIO63)	EALLOW	<a href="#">Go</a>
68h	GPBCSEL1	GPIO B Master Core Select (GPIO32 to GPIO39)	EALLOW	<a href="#">Go</a>
6Ah	GPBCSEL2	GPIO B Master Core Select (GPIO40 to GPIO47)	EALLOW	<a href="#">Go</a>
6Ch	GPBCSEL3	GPIO B Master Core Select (GPIO48 to GPIO55)	EALLOW	<a href="#">Go</a>
6Eh	GPBCSEL4	GPIO B Master Core Select (GPIO56 to GPIO63)	EALLOW	<a href="#">Go</a>
7Ch	GPBLOCK	GPIO B Lock Register (GPIO32 to GPIO63)	EALLOW	<a href="#">Go</a>

**Table 8-9. GPIO\_CTRL\_REGS Registers (continued)**

Offset	Acronym	Register Name	Write Protection	Section
7Eh	GPBCR	GPIO B Lock Commit Register (GPIO32 to GPIO63)	EALLOW	<a href="#">Go</a>
1C0h	GPHCTRL	GPIO H Qualification Sampling Period (GPIO224 to GPIO255)	EALLOW	<a href="#">Go</a>
1C2h	GPHQSEL1	GPIO H Qualification Type (GPIO224 to GPIO239)	EALLOW	<a href="#">Go</a>
1C4h	GPHQSEL2	GPIO H Qualification Type (GPIO240 to GPIO255)	EALLOW	<a href="#">Go</a>
1CCh	GPHPUD	GPIO H Pull-Up Disable (GPIO224 to GPIO255)	EALLOW	<a href="#">Go</a>
1D0h	GPHINV	GPIO H Input Inversion (GPIO224 to GPIO255)	EALLOW	<a href="#">Go</a>
1D4h	GPHAMSEL	GPIO H Analog Mode Select (GPIO224 to GPIO255)	EALLOW	<a href="#">Go</a>
1FCh	GPHLOCK	GPIO H Lock Register (GPIO224 to GPIO255)	EALLOW	<a href="#">Go</a>
1FEh	GPHCR	GPIO H Lock Commit Register (GPIO224 to GPIO255)	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 8-10](#) shows the codes that are used for access types in this section.

**Table 8-10. GPIO\_CTRL\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
WOnce	W Once	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.



### 8.7.2.1 GPACTRL Register (Offset = 0h) [reset = 0h]

GPACTRL is shown in [Figure 8-4](#) and described in [Table 8-11](#).

Return to the [Summary Table](#).

GPIO A Qualification Sampling Period (GPIO0 to GPIO31)

Each field in this register selects the qualification sampling period in SYSCLK cycles for eight GPIOs. The period is equal to 2 times the register field value.

0x00: Period = 0 SYSCLK cycles

0x01: Period = 2 SYSCLK cycles

0x02: Period = 4 SYSCLK cycles

...

0xFF: Period = 510 SYSCLK cycles

**Figure 8-4. GPACTRL 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
QUALPRD3								QUALPRD2								QUALPRD1								QUALPRD0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 8-11. GPACTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	QUALPRD3	R/W	0h	Qualification sampling period for GPIO24 to GPIO31 Reset type: SYSRSn
23-16	QUALPRD2	R/W	0h	Qualification sampling period for GPIO16 to GPIO23 Reset type: SYSRSn
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO8 to GPIO15 Reset type: SYSRSn
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO0 to GPIO7 Reset type: SYSRSn

### 8.7.2.2 GPAQSEL1 Register (Offset = 2h) [reset = 0h]

GPAQSEL1 is shown in [Figure 8-5](#) and described in [Table 8-12](#).

Return to the [Summary Table](#).

GPIO A Qualification Type (GPIO0 to GPIO15)

Each field in this register selects the input qualification type for one IO pin. The available types are:

- 0: Synchronous
- 1: 3-sample qualification
- 2: 6-sample qualification
- 3: Asynchronous

**Figure 8-5. GPAQSEL1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO15		GPIO14		GPIO13		GPIO12		GPIO11		GPIO10		GPIO9		GPIO8	
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	7	6	5	4	3	2	1	0
GPIO7		GPIO6		GPIO5		GPIO4		GPIO3		GPIO2		GPIO1		GPIO0	
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 8-12. GPAQSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO15	R/W	0h	Input qualification type for GPIO15 Reset type: SYSRSn
29-28	GPIO14	R/W	0h	Input qualification type for GPIO14 Reset type: SYSRSn
27-26	GPIO13	R/W	0h	Input qualification type for GPIO13 Reset type: SYSRSn
25-24	GPIO12	R/W	0h	Input qualification type for GPIO12 Reset type: SYSRSn
23-22	GPIO11	R/W	0h	Input qualification type for GPIO11 Reset type: SYSRSn
21-20	GPIO10	R/W	0h	Input qualification type for GPIO10 Reset type: SYSRSn
19-18	GPIO9	R/W	0h	Input qualification type for GPIO9 Reset type: SYSRSn
17-16	GPIO8	R/W	0h	Input qualification type for GPIO8 Reset type: SYSRSn
15-14	GPIO7	R/W	0h	Input qualification type for GPIO7 Reset type: SYSRSn
13-12	GPIO6	R/W	0h	Input qualification type for GPIO6 Reset type: SYSRSn
11-10	GPIO5	R/W	0h	Input qualification type for GPIO5 Reset type: SYSRSn
9-8	GPIO4	R/W	0h	Input qualification type for GPIO4 Reset type: SYSRSn
7-6	GPIO3	R/W	0h	Input qualification type for GPIO3 Reset type: SYSRSn
5-4	GPIO2	R/W	0h	Input qualification type for GPIO2 Reset type: SYSRSn

**Table 8-12. GPAQSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	GPIO1	R/W	0h	Input qualification type for GPIO1 Reset type: SYSRSn
1-0	GPIO0	R/W	0h	Input qualification type for GPIO0 Reset type: SYSRSn

### 8.7.2.3 GPAQSEL2 Register (Offset = 4h) [reset = 0h]

GPAQSEL2 is shown in [Figure 8-6](#) and described in [Table 8-13](#).

Return to the [Summary Table](#).

GPIO A Qualification Type (GPIO16 to GPIO31)

Each field in this register determines the input qualification type for one IO pin. The available types are:

- 0: Synchronous
- 1: 3-sample qualification
- 2: 6-sample qualification
- 3: Asynchronous

**Figure 8-6. GPAQSEL2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO31		GPIO30		GPIO29		GPIO28		GPIO27		GPIO26		GPIO25		GPIO24	
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	7	6	5	4	3	2	1	0
GPIO23		GPIO22		GPIO21		GPIO20		GPIO19		GPIO18		GPIO17		GPIO16	
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 8-13. GPAQSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO31	R/W	0h	Input qualification type for GPIO31 Reset type: SYSRSn
29-28	GPIO30	R/W	0h	Input qualification type for GPIO30 Reset type: SYSRSn
27-26	GPIO29	R/W	0h	Input qualification type for GPIO29 Reset type: SYSRSn
25-24	GPIO28	R/W	0h	Input qualification type for GPIO28 Reset type: SYSRSn
23-22	GPIO27	R/W	0h	Input qualification type for GPIO27 Reset type: SYSRSn
21-20	GPIO26	R/W	0h	Input qualification type for GPIO26 Reset type: SYSRSn
19-18	GPIO25	R/W	0h	Input qualification type for GPIO25 Reset type: SYSRSn
17-16	GPIO24	R/W	0h	Input qualification type for GPIO24 Reset type: SYSRSn
15-14	GPIO23	R/W	0h	Input qualification type for GPIO23 Reset type: SYSRSn
13-12	GPIO22	R/W	0h	Input qualification type for GPIO22 Reset type: SYSRSn
11-10	GPIO21	R/W	0h	Input qualification type for GPIO21 Reset type: SYSRSn
9-8	GPIO20	R/W	0h	Input qualification type for GPIO20 Reset type: SYSRSn
7-6	GPIO19	R/W	0h	Input qualification type for GPIO19 Reset type: SYSRSn
5-4	GPIO18	R/W	0h	Input qualification type for GPIO18 Reset type: SYSRSn

**Table 8-13. GPAQSEL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	GPIO17	R/W	0h	Input qualification type for GPIO17 Reset type: SYSRSn
1-0	GPIO16	R/W	0h	Input qualification type for GPIO16 Reset type: SYSRSn

### 8.7.2.4 GPAMUX1 Register (Offset = 6h) [reset = 0h]

GPAMUX1 is shown in [Figure 8-7](#) and described in [Table 8-14](#).

Return to the [Summary Table](#).

GPIO A Peripheral Mux (GPIO0 to GPIO15)

Each field in this register determines part of the GPIO mux configuration for one IO pin. A value of 0x0, 0x4, 0x8, or 0xC configures the pin as a general-purpose IO. All other values select a peripheral to control the pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using this register before changing the corresponding field in the GPAGMUX1 register.

**Figure 8-7. GPAMUX1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO15		GPIO14		GPIO13		GPIO12		GPIO11		GPIO10		GPIO9		GPIO8	
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	7	6	5	4	3	2	1	0
GPIO7		GPIO6		GPIO5		GPIO4		GPIO3		GPIO2		GPIO1		GPIO0	
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 8-14. GPAMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO15	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO15 Reset type: SYSRSn
29-28	GPIO14	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO14 Reset type: SYSRSn
27-26	GPIO13	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO13 Reset type: SYSRSn
25-24	GPIO12	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO12 Reset type: SYSRSn
23-22	GPIO11	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO11 Reset type: SYSRSn
21-20	GPIO10	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO10 Reset type: SYSRSn
19-18	GPIO9	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO9 Reset type: SYSRSn
17-16	GPIO8	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO8 Reset type: SYSRSn
15-14	GPIO7	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO7 Reset type: SYSRSn
13-12	GPIO6	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO6 Reset type: SYSRSn
11-10	GPIO5	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO5 Reset type: SYSRSn
9-8	GPIO4	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO4 Reset type: SYSRSn
7-6	GPIO3	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO3 Reset type: SYSRSn
5-4	GPIO2	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO2 Reset type: SYSRSn
3-2	GPIO1	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO1 Reset type: SYSRSn

**Table 8-14. GPAMUX1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GPIO0	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO0 Reset type: SYSRSn

### 8.7.2.5 GPAMUX2 Register (Offset = 8h) [reset = 0h]

GPAMUX2 is shown in [Figure 8-8](#) and described in [Table 8-15](#).

Return to the [Summary Table](#).

GPIO A Peripheral Mux (GPIO16 to GPIO31)

Each field in this register determines part of the GPIO mux configuration for one IO pin. A value of 0x0, 0x4, 0x8, or 0xC configures the pin as a general-purpose IO. All other values select a peripheral to control the pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using this register before changing the corresponding field in the GPAGMUX2 register.

**Figure 8-8. GPAMUX2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO31		GPIO30		GPIO29		GPIO28		GPIO27		GPIO26		GPIO25		GPIO24	
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	7	6	5	4	3	2	1	0
GPIO23		GPIO22		GPIO21		GPIO20		GPIO19		GPIO18		GPIO17		GPIO16	
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 8-15. GPAMUX2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO31	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO31 Reset type: SYSRSn
29-28	GPIO30	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO30 Reset type: SYSRSn
27-26	GPIO29	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO29 Reset type: SYSRSn
25-24	GPIO28	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO28 Reset type: SYSRSn
23-22	GPIO27	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO27 Reset type: SYSRSn
21-20	GPIO26	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO26 Reset type: SYSRSn
19-18	GPIO25	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO25 Reset type: SYSRSn
17-16	GPIO24	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO24 Reset type: SYSRSn
15-14	GPIO23	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO23 Reset type: SYSRSn
13-12	GPIO22	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO22 Reset type: SYSRSn
11-10	GPIO21	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO21 Reset type: SYSRSn
9-8	GPIO20	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO20 Reset type: SYSRSn
7-6	GPIO19	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO19 Reset type: SYSRSn
5-4	GPIO18	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO18 Reset type: SYSRSn
3-2	GPIO17	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO17 Reset type: SYSRSn



**Table 8-15. GPAMUX2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GPIO16	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO16 Reset type: SYSRSn

### 8.7.2.6 GPADIR Register (Offset = Ah) [reset = 0h]

GPADIR is shown in [Figure 8-9](#) and described in [Table 8-16](#).

Return to the [Summary Table](#).

GPIO A Direction (GPIO0 to GPIO31)

Each field in this register selects the direction of one IO pin in GPIO mode. If the pin is not configured as a general-purpose IO, this register has no effect.

0: The pin is an input

1: The pin is an output

**Figure 8-9. GPADIR Register**

31		30		29		28		27		26		25		24	
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24	GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16	GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-16. GPADIR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Data direction for GPIO31 Reset type: SYSRSn
30	GPIO30	R/W	0h	Data direction for GPIO30 Reset type: SYSRSn
29	GPIO29	R/W	0h	Data direction for GPIO29 Reset type: SYSRSn
28	GPIO28	R/W	0h	Data direction for GPIO28 Reset type: SYSRSn
27	GPIO27	R/W	0h	Data direction for GPIO27 Reset type: SYSRSn
26	GPIO26	R/W	0h	Data direction for GPIO26 Reset type: SYSRSn
25	GPIO25	R/W	0h	Data direction for GPIO25 Reset type: SYSRSn
24	GPIO24	R/W	0h	Data direction for GPIO24 Reset type: SYSRSn
23	GPIO23	R/W	0h	Data direction for GPIO23 Reset type: SYSRSn
22	GPIO22	R/W	0h	Data direction for GPIO22 Reset type: SYSRSn
21	GPIO21	R/W	0h	Data direction for GPIO21 Reset type: SYSRSn
20	GPIO20	R/W	0h	Data direction for GPIO20 Reset type: SYSRSn

**Table 8-16. GPADIR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	GPIO19	R/W	0h	Data direction for GPIO19 Reset type: SYSRSn
18	GPIO18	R/W	0h	Data direction for GPIO18 Reset type: SYSRSn
17	GPIO17	R/W	0h	Data direction for GPIO17 Reset type: SYSRSn
16	GPIO16	R/W	0h	Data direction for GPIO16 Reset type: SYSRSn
15	GPIO15	R/W	0h	Data direction for GPIO15 Reset type: SYSRSn
14	GPIO14	R/W	0h	Data direction for GPIO14 Reset type: SYSRSn
13	GPIO13	R/W	0h	Data direction for GPIO13 Reset type: SYSRSn
12	GPIO12	R/W	0h	Data direction for GPIO12 Reset type: SYSRSn
11	GPIO11	R/W	0h	Data direction for GPIO11 Reset type: SYSRSn
10	GPIO10	R/W	0h	Data direction for GPIO10 Reset type: SYSRSn
9	GPIO9	R/W	0h	Data direction for GPIO9 Reset type: SYSRSn
8	GPIO8	R/W	0h	Data direction for GPIO8 Reset type: SYSRSn
7	GPIO7	R/W	0h	Data direction for GPIO7 Reset type: SYSRSn
6	GPIO6	R/W	0h	Data direction for GPIO6 Reset type: SYSRSn
5	GPIO5	R/W	0h	Data direction for GPIO5 Reset type: SYSRSn
4	GPIO4	R/W	0h	Data direction for GPIO4 Reset type: SYSRSn
3	GPIO3	R/W	0h	Data direction for GPIO3 Reset type: SYSRSn
2	GPIO2	R/W	0h	Data direction for GPIO2 Reset type: SYSRSn
1	GPIO1	R/W	0h	Data direction for GPIO1 Reset type: SYSRSn
0	GPIO0	R/W	0h	Data direction for GPIO0 Reset type: SYSRSn

### 8.7.2.7 GPAPUD Register (Offset = Ch) [reset = FFFFFFFFh]

GPAPUD is shown in [Figure 8-10](#) and described in [Table 8-17](#).

Return to the [Summary Table](#).

GPIO A Pull-Up Disable (GPIO0 to GPIO31)

Each field in this register selects the state of the internal pull-up resistor for a single IO pin.

0: Pull-up enabled

1: Pull-up disabled

**Figure 8-10. GPAPUD Register**

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h

**Table 8-17. GPAPUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	1h	Pull-up disable for GPIO31 Reset type: SYSRSn
30	GPIO30	R/W	1h	Pull-up disable for GPIO30 Reset type: SYSRSn
29	GPIO29	R/W	1h	Pull-up disable for GPIO29 Reset type: SYSRSn
28	GPIO28	R/W	1h	Pull-up disable for GPIO28 Reset type: SYSRSn
27	GPIO27	R/W	1h	Pull-up disable for GPIO27 Reset type: SYSRSn
26	GPIO26	R/W	1h	Pull-up disable for GPIO26 Reset type: SYSRSn
25	GPIO25	R/W	1h	Pull-up disable for GPIO25 Reset type: SYSRSn
24	GPIO24	R/W	1h	Pull-up disable for GPIO24 Reset type: SYSRSn
23	GPIO23	R/W	1h	Pull-up disable for GPIO23 Reset type: SYSRSn
22	GPIO22	R/W	1h	Pull-up disable for GPIO22 Reset type: SYSRSn
21	GPIO21	R/W	1h	Pull-up disable for GPIO21 Reset type: SYSRSn
20	GPIO20	R/W	1h	Pull-up disable for GPIO20 Reset type: SYSRSn

**Table 8-17. GPAPUD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	GPIO19	R/W	1h	Pull-up disable for GPIO19 Reset type: SYSRSn
18	GPIO18	R/W	1h	Pull-up disable for GPIO18 Reset type: SYSRSn
17	GPIO17	R/W	1h	Pull-up disable for GPIO17 Reset type: SYSRSn
16	GPIO16	R/W	1h	Pull-up disable for GPIO16 Reset type: SYSRSn
15	GPIO15	R/W	1h	Pull-up disable for GPIO15 Reset type: SYSRSn
14	GPIO14	R/W	1h	Pull-up disable for GPIO14 Reset type: SYSRSn
13	GPIO13	R/W	1h	Pull-up disable for GPIO13 Reset type: SYSRSn
12	GPIO12	R/W	1h	Pull-up disable for GPIO12 Reset type: SYSRSn
11	GPIO11	R/W	1h	Pull-up disable for GPIO11 Reset type: SYSRSn
10	GPIO10	R/W	1h	Pull-up disable for GPIO10 Reset type: SYSRSn
9	GPIO9	R/W	1h	Pull-up disable for GPIO9 Reset type: SYSRSn
8	GPIO8	R/W	1h	Pull-up disable for GPIO8 Reset type: SYSRSn
7	GPIO7	R/W	1h	Pull-up disable for GPIO7 Reset type: SYSRSn
6	GPIO6	R/W	1h	Pull-up disable for GPIO6 Reset type: SYSRSn
5	GPIO5	R/W	1h	Pull-up disable for GPIO5 Reset type: SYSRSn
4	GPIO4	R/W	1h	Pull-up disable for GPIO4 Reset type: SYSRSn
3	GPIO3	R/W	1h	Pull-up disable for GPIO3 Reset type: SYSRSn
2	GPIO2	R/W	1h	Pull-up disable for GPIO2 Reset type: SYSRSn
1	GPIO1	R/W	1h	Pull-up disable for GPIO1 Reset type: SYSRSn
0	GPIO0	R/W	1h	Pull-up disable for GPIO0 Reset type: SYSRSn

### 8.7.2.8 GPAINV Register (Offset = 10h) [reset = 0h]

GPAINV is shown in [Figure 8-11](#) and described in [Table 8-18](#).

Return to the [Summary Table](#).

GPIO A Input Inversion (GPIO0 to GPIO31)

Each field in this register selects whether the input value of one IO pin passes through an inverter.

0: The input is not inverted

1: The input is inverted

**Figure 8-11. GPAINV Register**

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
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
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
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
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
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 8-18. GPAINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Input inversion for GPIO31 Reset type: SYSRSn
30	GPIO30	R/W	0h	Input inversion for GPIO30 Reset type: SYSRSn
29	GPIO29	R/W	0h	Input inversion for GPIO29 Reset type: SYSRSn
28	GPIO28	R/W	0h	Input inversion for GPIO28 Reset type: SYSRSn
27	GPIO27	R/W	0h	Input inversion for GPIO27 Reset type: SYSRSn
26	GPIO26	R/W	0h	Input inversion for GPIO26 Reset type: SYSRSn
25	GPIO25	R/W	0h	Input inversion for GPIO25 Reset type: SYSRSn
24	GPIO24	R/W	0h	Input inversion for GPIO24 Reset type: SYSRSn
23	GPIO23	R/W	0h	Input inversion for GPIO23 Reset type: SYSRSn
22	GPIO22	R/W	0h	Input inversion for GPIO22 Reset type: SYSRSn
21	GPIO21	R/W	0h	Input inversion for GPIO21 Reset type: SYSRSn
20	GPIO20	R/W	0h	Input inversion for GPIO20 Reset type: SYSRSn

**Table 8-18. GPAINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	GPIO19	R/W	0h	Input inversion for GPIO19 Reset type: SYSRSn
18	GPIO18	R/W	0h	Input inversion for GPIO18 Reset type: SYSRSn
17	GPIO17	R/W	0h	Input inversion for GPIO17 Reset type: SYSRSn
16	GPIO16	R/W	0h	Input inversion for GPIO16 Reset type: SYSRSn
15	GPIO15	R/W	0h	Input inversion for GPIO15 Reset type: SYSRSn
14	GPIO14	R/W	0h	Input inversion for GPIO14 Reset type: SYSRSn
13	GPIO13	R/W	0h	Input inversion for GPIO13 Reset type: SYSRSn
12	GPIO12	R/W	0h	Input inversion for GPIO12 Reset type: SYSRSn
11	GPIO11	R/W	0h	Input inversion for GPIO11 Reset type: SYSRSn
10	GPIO10	R/W	0h	Input inversion for GPIO10 Reset type: SYSRSn
9	GPIO9	R/W	0h	Input inversion for GPIO9 Reset type: SYSRSn
8	GPIO8	R/W	0h	Input inversion for GPIO8 Reset type: SYSRSn
7	GPIO7	R/W	0h	Input inversion for GPIO7 Reset type: SYSRSn
6	GPIO6	R/W	0h	Input inversion for GPIO6 Reset type: SYSRSn
5	GPIO5	R/W	0h	Input inversion for GPIO5 Reset type: SYSRSn
4	GPIO4	R/W	0h	Input inversion for GPIO4 Reset type: SYSRSn
3	GPIO3	R/W	0h	Input inversion for GPIO3 Reset type: SYSRSn
2	GPIO2	R/W	0h	Input inversion for GPIO2 Reset type: SYSRSn
1	GPIO1	R/W	0h	Input inversion for GPIO1 Reset type: SYSRSn
0	GPIO0	R/W	0h	Input inversion for GPIO0 Reset type: SYSRSn

### 8.7.2.9 GPAODR Register (Offset = 12h) [reset = 0h]

GPAODR is shown in [Figure 8-12](#) and described in [Table 8-19](#).

Return to the [Summary Table](#).

GPIO A Open Drain Output Mode (GPIO0 to GPIO31)

Each field in this register selects between push-pull mode and open-drain mode for one general-purpose output pin. In both modes, writing a 0 to the output data latch drives the pin low. In push-pull mode, writing a 1 to the output data latch drives the pin high. In open-drain mode, it tri-states the output buffer.

0: Push-pull output

1: Open-drain output

**Figure 8-12. GPAODR Register**

31		30		29		28		27		26		25		24	
GPIO31		GPIO30		GPIO29		GPIO28		GPIO27		GPIO26		GPIO25		GPIO24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23		22		21		20		19		18		17		16	
GPIO23		GPIO22		GPIO21		GPIO20		GPIO19		GPIO18		GPIO17		GPIO16	
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	
GPIO15		GPIO14		GPIO13		GPIO12		GPIO11		GPIO10		GPIO9		GPIO8	
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	
GPIO7		GPIO6		GPIO5		GPIO4		GPIO3		GPIO2		GPIO1		GPIO0	
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 8-19. GPAODR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Open-drain output mode for GPIO31 Reset type: SYSRSn
30	GPIO30	R/W	0h	Open-drain output mode for GPIO30 Reset type: SYSRSn
29	GPIO29	R/W	0h	Open-drain output mode for GPIO29 Reset type: SYSRSn
28	GPIO28	R/W	0h	Open-drain output mode for GPIO28 Reset type: SYSRSn
27	GPIO27	R/W	0h	Open-drain output mode for GPIO27 Reset type: SYSRSn
26	GPIO26	R/W	0h	Open-drain output mode for GPIO26 Reset type: SYSRSn
25	GPIO25	R/W	0h	Open-drain output mode for GPIO25 Reset type: SYSRSn
24	GPIO24	R/W	0h	Open-drain output mode for GPIO24 Reset type: SYSRSn
23	GPIO23	R/W	0h	Open-drain output mode for GPIO23 Reset type: SYSRSn
22	GPIO22	R/W	0h	Open-drain output mode for GPIO22 Reset type: SYSRSn
21	GPIO21	R/W	0h	Open-drain output mode for GPIO21 Reset type: SYSRSn



**Table 8-19. GPAODR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	GPIO20	R/W	0h	Open-drain output mode for GPIO20 Reset type: SYSRSn
19	GPIO19	R/W	0h	Open-drain output mode for GPIO19 Reset type: SYSRSn
18	GPIO18	R/W	0h	Open-drain output mode for GPIO18 Reset type: SYSRSn
17	GPIO17	R/W	0h	Open-drain output mode for GPIO17 Reset type: SYSRSn
16	GPIO16	R/W	0h	Open-drain output mode for GPIO16 Reset type: SYSRSn
15	GPIO15	R/W	0h	Open-drain output mode for GPIO15 Reset type: SYSRSn
14	GPIO14	R/W	0h	Open-drain output mode for GPIO14 Reset type: SYSRSn
13	GPIO13	R/W	0h	Open-drain output mode for GPIO13 Reset type: SYSRSn
12	GPIO12	R/W	0h	Open-drain output mode for GPIO12 Reset type: SYSRSn
11	GPIO11	R/W	0h	Open-drain output mode for GPIO11 Reset type: SYSRSn
10	GPIO10	R/W	0h	Open-drain output mode for GPIO10 Reset type: SYSRSn
9	GPIO9	R/W	0h	Open-drain output mode for GPIO9 Reset type: SYSRSn
8	GPIO8	R/W	0h	Open-drain output mode for GPIO8 Reset type: SYSRSn
7	GPIO7	R/W	0h	Open-drain output mode for GPIO7 Reset type: SYSRSn
6	GPIO6	R/W	0h	Open-drain output mode for GPIO6 Reset type: SYSRSn
5	GPIO5	R/W	0h	Open-drain output mode for GPIO5 Reset type: SYSRSn
4	GPIO4	R/W	0h	Open-drain output mode for GPIO4 Reset type: SYSRSn
3	GPIO3	R/W	0h	Open-drain output mode for GPIO3 Reset type: SYSRSn
2	GPIO2	R/W	0h	Open-drain output mode for GPIO2 Reset type: SYSRSn
1	GPIO1	R/W	0h	Open-drain output mode for GPIO1 Reset type: SYSRSn
0	GPIO0	R/W	0h	Open-drain output mode for GPIO0 Reset type: SYSRSn

### 8.7.2.10 GPAAMSEL Register (Offset = 14h) [reset = 0h]

GPAAMSEL is shown in [Figure 8-13](#) and described in [Table 8-20](#).

Return to the [Summary Table](#).

GPIO A Analog Mode Select (GPIO0 to GPIO31)

Each field in this register selects between analog and digital functionality for one IO pin.

- 0: Digital mode
- 1: Analog mode

**Figure 8-13. GPAAMSEL Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
23	22	21	20	19	18	17	16
GPIO23	GPIO22	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R/W-0h	R/W-0h						
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED

**Table 8-20. GPAAMSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	GPIO23	R/W	0h	Analog mode select for GPIO23 Reset type: SYSRSn
22	GPIO22	R/W	0h	Analog mode select for GPIO22 Reset type: SYSRSn
21	RESERVED	R/W	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved

**Table 8-20. GPAAMSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

### 8.7.2.11 GPAGMUX1 Register (Offset = 20h) [reset = 0h]

GPAGMUX1 is shown in [Figure 8-14](#) and described in [Table 8-21](#).

Return to the [Summary Table](#).

GPIO A Peripheral Group Mux (GPIO0 to GPIO15)

Each field in this register determines part of the GPIO mux configuration for one IO pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using the GPAMUX1 register before changing their configuration in this register.

**Figure 8-14. GPAGMUX1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO15		GPIO14		GPIO13		GPIO12		GPIO11		GPIO10		GPIO9		GPIO8	
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	7	6	5	4	3	2	1	0
GPIO7		GPIO6		GPIO5		GPIO4		GPIO3		GPIO2		GPIO1		GPIO0	
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 8-21. GPAGMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO15	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO15 Reset type: SYSRSn
29-28	GPIO14	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO14 Reset type: SYSRSn
27-26	GPIO13	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO13 Reset type: SYSRSn
25-24	GPIO12	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO12 Reset type: SYSRSn
23-22	GPIO11	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO11 Reset type: SYSRSn
21-20	GPIO10	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO10 Reset type: SYSRSn
19-18	GPIO9	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO9 Reset type: SYSRSn
17-16	GPIO8	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO8 Reset type: SYSRSn
15-14	GPIO7	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO7 Reset type: SYSRSn
13-12	GPIO6	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO6 Reset type: SYSRSn
11-10	GPIO5	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO5 Reset type: SYSRSn
9-8	GPIO4	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO4 Reset type: SYSRSn
7-6	GPIO3	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO3 Reset type: SYSRSn
5-4	GPIO2	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO2 Reset type: SYSRSn
3-2	GPIO1	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO1 Reset type: SYSRSn

**Table 8-21. GPAGMUX1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GPIO0	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO0 Reset type: SYSRSn

### 8.7.2.12 GPAGMUX2 Register (Offset = 22h) [reset = 0h]

GPAGMUX2 is shown in [Figure 8-15](#) and described in [Table 8-22](#).

Return to the [Summary Table](#).

GPIO A Peripheral Group Mux (GPIO16 to GPIO31)

Each field in this register determines part of the GPIO mux configuration for a single IO pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using the GPAMUX2 register before changing their configuration in this register.

**Figure 8-15. GPAGMUX2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO31		GPIO30		GPIO29		GPIO28		GPIO27		GPIO26		GPIO25		GPIO24	
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	7	6	5	4	3	2	1	0
GPIO23		GPIO22		GPIO21		GPIO20		GPIO19		GPIO18		GPIO17		GPIO16	
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 8-22. GPAGMUX2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO31	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO31 Reset type: SYSRSn
29-28	GPIO30	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO30 Reset type: SYSRSn
27-26	GPIO29	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO29 Reset type: SYSRSn
25-24	GPIO28	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO28 Reset type: SYSRSn
23-22	GPIO27	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO27 Reset type: SYSRSn
21-20	GPIO26	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO26 Reset type: SYSRSn
19-18	GPIO25	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO25 Reset type: SYSRSn
17-16	GPIO24	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO24 Reset type: SYSRSn
15-14	GPIO23	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO23 Reset type: SYSRSn
13-12	GPIO22	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO22 Reset type: SYSRSn
11-10	GPIO21	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO21 Reset type: SYSRSn
9-8	GPIO20	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO20 Reset type: SYSRSn
7-6	GPIO19	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO19 Reset type: SYSRSn
5-4	GPIO18	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO18 Reset type: SYSRSn
3-2	GPIO17	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO17 Reset type: SYSRSn

**Table 8-22. GPAGMUX2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GPIO16	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO16 Reset type: SYSRSn

### 8.7.2.13 GPACSEL1 Register (Offset = 28h) [reset = 0h]

GPACSEL1 is shown in [Figure 8-16](#) and described in [Table 8-23](#).

Return to the [Summary Table](#).

GPIO A Master Core Select (GPIO0 to GPIO7)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPADAT, GPASET, GPACLEAR, and GPATOGGLE registers.

0x0: CPU is the master

0x1: CLA is the master

0x2 - 0xF: Reserved

**Figure 8-16. GPACSEL1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO7				GPIO6				GPIO5				GPIO4			
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
GPIO3				GPIO2				GPIO1				GPIO0			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

**Table 8-23. GPACSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	GPIO7	R/W	0h	Master core select for GPIO7 Reset type: SYSRSn
27-24	GPIO6	R/W	0h	Master core select for GPIO6 Reset type: SYSRSn
23-20	GPIO5	R/W	0h	Master core select for GPIO5 Reset type: SYSRSn
19-16	GPIO4	R/W	0h	Master core select for GPIO4 Reset type: SYSRSn
15-12	GPIO3	R/W	0h	Master core select for GPIO3 Reset type: SYSRSn
11-8	GPIO2	R/W	0h	Master core select for GPIO2 Reset type: SYSRSn
7-4	GPIO1	R/W	0h	Master core select for GPIO1 Reset type: SYSRSn
3-0	GPIO0	R/W	0h	Master core select for GPIO0 Reset type: SYSRSn



### 8.7.2.14 GPACSEL2 Register (Offset = 2Ah) [reset = 0h]

GPACSEL2 is shown in [Figure 8-17](#) and described in [Table 8-24](#).

Return to the [Summary Table](#).

GPIO A Master Core Select (GPIO8 to GPIO15)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPADAT, GPASET, GPACLEAR, and GPATOGGLE registers.

0x0: CPU is the master

0x1: CLA is the master

0x2 - 0xF: Reserved

**Figure 8-17. GPACSEL2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO15				GPIO14				GPIO13				GPIO12			
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
GPIO11				GPIO10				GPIO9				GPIO8			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

**Table 8-24. GPACSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	GPIO15	R/W	0h	Master core select for GPIO15 Reset type: SYSRSn
27-24	GPIO14	R/W	0h	Master core select for GPIO14 Reset type: SYSRSn
23-20	GPIO13	R/W	0h	Master core select for GPIO13 Reset type: SYSRSn
19-16	GPIO12	R/W	0h	Master core select for GPIO12 Reset type: SYSRSn
15-12	GPIO11	R/W	0h	Master core select for GPIO11 Reset type: SYSRSn
11-8	GPIO10	R/W	0h	Master core select for GPIO10 Reset type: SYSRSn
7-4	GPIO9	R/W	0h	Master core select for GPIO9 Reset type: SYSRSn
3-0	GPIO8	R/W	0h	Master core select for GPIO8 Reset type: SYSRSn

### 8.7.2.15 GPACSEL3 Register (Offset = 2Ch) [reset = 0h]

GPACSEL3 is shown in [Figure 8-18](#) and described in [Table 8-25](#).

Return to the [Summary Table](#).

GPIO A Master Core Select (GPIO16 to GPIO23)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPADAT, GPASET, GPACLEAR, and GPATOGGLE registers.

0x0: CPU is the master

0x1: CLA is the master

0x2 - 0xF: Reserved

**Figure 8-18. GPACSEL3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO23				GPIO22				GPIO21				GPIO20			
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
GPIO19				GPIO18				GPIO17				GPIO16			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

**Table 8-25. GPACSEL3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	GPIO23	R/W	0h	Master core select for GPIO23 Reset type: SYSRSn
27-24	GPIO22	R/W	0h	Master core select for GPIO22 Reset type: SYSRSn
23-20	GPIO21	R/W	0h	Master core select for GPIO21 Reset type: SYSRSn
19-16	GPIO20	R/W	0h	Master core select for GPIO20 Reset type: SYSRSn
15-12	GPIO19	R/W	0h	Master core select for GPIO19 Reset type: SYSRSn
11-8	GPIO18	R/W	0h	Master core select for GPIO18 Reset type: SYSRSn
7-4	GPIO17	R/W	0h	Master core select for GPIO17 Reset type: SYSRSn
3-0	GPIO16	R/W	0h	Master core select for GPIO16 Reset type: SYSRSn

### 8.7.2.16 GPACSEL4 Register (Offset = 2Eh) [reset = 0h]

GPACSEL4 is shown in [Figure 8-19](#) and described in [Table 8-26](#).

Return to the [Summary Table](#).

GPIO A Master Core Select (GPIO24 to GPIO31)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPADAT, GPASET, GPACLEAR, and GPATOGGLE registers.

0x0: CPU is the master

0x1: CLA is the master

0x2 - 0xF: Reserved

**Figure 8-19. GPACSEL4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO31				GPIO30				GPIO29				GPIO28			
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
GPIO27				GPIO26				GPIO25				GPIO24			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

**Table 8-26. GPACSEL4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	GPIO31	R/W	0h	Master core select for GPIO31 Reset type: SYSRSn
27-24	GPIO30	R/W	0h	Master core select for GPIO30 Reset type: SYSRSn
23-20	GPIO29	R/W	0h	Master core select for GPIO29 Reset type: SYSRSn
19-16	GPIO28	R/W	0h	Master core select for GPIO28 Reset type: SYSRSn
15-12	GPIO27	R/W	0h	Master core select for GPIO27 Reset type: SYSRSn
11-8	GPIO26	R/W	0h	Master core select for GPIO26 Reset type: SYSRSn
7-4	GPIO25	R/W	0h	Master core select for GPIO25 Reset type: SYSRSn
3-0	GPIO24	R/W	0h	Master core select for GPIO24 Reset type: SYSRSn

### 8.7.2.17 GPALOCK Register (Offset = 3Ch) [reset = 0h]

GPALOCK is shown in [Figure 8-20](#) and described in [Table 8-27](#).

Return to the [Summary Table](#).

GPIO A Lock Register (GPIO0 to GPIO31)

Each field in this register locks one IO pin's configuration. This blocks writes to the corresponding bits in the GPAMUXn, GPAIDR, GPAINV, GPAODR, GPAGMUXn, and GPACSELn registers.

0: Pin configuration is unlocked

1: Pin configuration is locked

**Figure 8-20. GPALOCK Register**

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
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
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
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
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
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 8-27. GPALOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Configuration lock for GPIO31 Reset type: SYSRSn
30	GPIO30	R/W	0h	Configuration lock for GPIO30 Reset type: SYSRSn
29	GPIO29	R/W	0h	Configuration lock for GPIO29 Reset type: SYSRSn
28	GPIO28	R/W	0h	Configuration lock for GPIO28 Reset type: SYSRSn
27	GPIO27	R/W	0h	Configuration lock for GPIO27 Reset type: SYSRSn
26	GPIO26	R/W	0h	Configuration lock for GPIO26 Reset type: SYSRSn
25	GPIO25	R/W	0h	Configuration lock for GPIO25 Reset type: SYSRSn
24	GPIO24	R/W	0h	Configuration lock for GPIO24 Reset type: SYSRSn
23	GPIO23	R/W	0h	Configuration lock for GPIO23 Reset type: SYSRSn
22	GPIO22	R/W	0h	Configuration lock for GPIO22 Reset type: SYSRSn
21	GPIO21	R/W	0h	Configuration lock for GPIO21 Reset type: SYSRSn
20	GPIO20	R/W	0h	Configuration lock for GPIO20 Reset type: SYSRSn

**Table 8-27. GPALOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	GPIO19	R/W	0h	Configuration lock for GPIO19 Reset type: SYSRSn
18	GPIO18	R/W	0h	Configuration lock for GPIO18 Reset type: SYSRSn
17	GPIO17	R/W	0h	Configuration lock for GPIO17 Reset type: SYSRSn
16	GPIO16	R/W	0h	Configuration lock for GPIO16 Reset type: SYSRSn
15	GPIO15	R/W	0h	Configuration lock for GPIO15 Reset type: SYSRSn
14	GPIO14	R/W	0h	Configuration lock for GPIO14 Reset type: SYSRSn
13	GPIO13	R/W	0h	Configuration lock for GPIO13 Reset type: SYSRSn
12	GPIO12	R/W	0h	Configuration lock for GPIO12 Reset type: SYSRSn
11	GPIO11	R/W	0h	Configuration lock for GPIO11 Reset type: SYSRSn
10	GPIO10	R/W	0h	Configuration lock for GPIO10 Reset type: SYSRSn
9	GPIO9	R/W	0h	Configuration lock for GPIO9 Reset type: SYSRSn
8	GPIO8	R/W	0h	Configuration lock for GPIO8 Reset type: SYSRSn
7	GPIO7	R/W	0h	Configuration lock for GPIO7 Reset type: SYSRSn
6	GPIO6	R/W	0h	Configuration lock for GPIO6 Reset type: SYSRSn
5	GPIO5	R/W	0h	Configuration lock for GPIO5 Reset type: SYSRSn
4	GPIO4	R/W	0h	Configuration lock for GPIO4 Reset type: SYSRSn
3	GPIO3	R/W	0h	Configuration lock for GPIO3 Reset type: SYSRSn
2	GPIO2	R/W	0h	Configuration lock for GPIO2 Reset type: SYSRSn
1	GPIO1	R/W	0h	Configuration lock for GPIO1 Reset type: SYSRSn
0	GPIO0	R/W	0h	Configuration lock for GPIO0 Reset type: SYSRSn

### 8.7.2.18 GPACR Register (Offset = 3Eh) [reset = 0h]

GPACR is shown in [Figure 8-21](#) and described in [Table 8-28](#).

Return to the [Summary Table](#).

GPIO A Lock Commit Register (GPIO0 to GPIO31)

Each field in this register blocks writes to one IO pin's GPALOCK bit. Once set, a lock commit can only be cleared by a reset.

0: Pin configuration lock is unlocked

1: Pin configuration lock is locked

**Figure 8-21. GPACR Register**

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h	R/WSonce-0h

**Table 8-28. GPACR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/WSonce	0h	Configuration lock commit for GPIO31 Reset type: SYSRSn
30	GPIO30	R/WSonce	0h	Configuration lock commit for GPIO30 Reset type: SYSRSn
29	GPIO29	R/WSonce	0h	Configuration lock commit for GPIO29 Reset type: SYSRSn
28	GPIO28	R/WSonce	0h	Configuration lock commit for GPIO28 Reset type: SYSRSn
27	GPIO27	R/WSonce	0h	Configuration lock commit for GPIO27 Reset type: SYSRSn
26	GPIO26	R/WSonce	0h	Configuration lock commit for GPIO26 Reset type: SYSRSn
25	GPIO25	R/WSonce	0h	Configuration lock commit for GPIO25 Reset type: SYSRSn
24	GPIO24	R/WSonce	0h	Configuration lock commit for GPIO24 Reset type: SYSRSn
23	GPIO23	R/WSonce	0h	Configuration lock commit for GPIO23 Reset type: SYSRSn
22	GPIO22	R/WSonce	0h	Configuration lock commit for GPIO22 Reset type: SYSRSn
21	GPIO21	R/WSonce	0h	Configuration lock commit for GPIO21 Reset type: SYSRSn
20	GPIO20	R/WSonce	0h	Configuration lock commit for GPIO20 Reset type: SYSRSn

**Table 8-28. GPACR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	GPIO19	R/WOnce	0h	Configuration lock commit for GPIO19 Reset type: SYSRSn
18	GPIO18	R/WOnce	0h	Configuration lock commit for GPIO18 Reset type: SYSRSn
17	GPIO17	R/WOnce	0h	Configuration lock commit for GPIO17 Reset type: SYSRSn
16	GPIO16	R/WOnce	0h	Configuration lock commit for GPIO16 Reset type: SYSRSn
15	GPIO15	R/WOnce	0h	Configuration lock commit for GPIO15 Reset type: SYSRSn
14	GPIO14	R/WOnce	0h	Configuration lock commit for GPIO14 Reset type: SYSRSn
13	GPIO13	R/WOnce	0h	Configuration lock commit for GPIO13 Reset type: SYSRSn
12	GPIO12	R/WOnce	0h	Configuration lock commit for GPIO12 Reset type: SYSRSn
11	GPIO11	R/WOnce	0h	Configuration lock commit for GPIO11 Reset type: SYSRSn
10	GPIO10	R/WOnce	0h	Configuration lock commit for GPIO10 Reset type: SYSRSn
9	GPIO9	R/WOnce	0h	Configuration lock commit for GPIO9 Reset type: SYSRSn
8	GPIO8	R/WOnce	0h	Configuration lock commit for GPIO8 Reset type: SYSRSn
7	GPIO7	R/WOnce	0h	Configuration lock commit for GPIO7 Reset type: SYSRSn
6	GPIO6	R/WOnce	0h	Configuration lock commit for GPIO6 Reset type: SYSRSn
5	GPIO5	R/WOnce	0h	Configuration lock commit for GPIO5 Reset type: SYSRSn
4	GPIO4	R/WOnce	0h	Configuration lock commit for GPIO4 Reset type: SYSRSn
3	GPIO3	R/WOnce	0h	Configuration lock commit for GPIO3 Reset type: SYSRSn
2	GPIO2	R/WOnce	0h	Configuration lock commit for GPIO2 Reset type: SYSRSn
1	GPIO1	R/WOnce	0h	Configuration lock commit for GPIO1 Reset type: SYSRSn
0	GPIO0	R/WOnce	0h	Configuration lock commit for GPIO0 Reset type: SYSRSn

### 8.7.2.19 GPBCTRL Register (Offset = 40h) [reset = 0h]

GPBCTRL is shown in [Figure 8-22](#) and described in [Table 8-29](#).

Return to the [Summary Table](#).

GPIO B Qualification Sampling Period (GPIO32 to GPIO63)

Each field in this register selects the qualification sampling period in SYSCLK cycles for eight GPIOs. The period is equal to 2 times the register field value.

0x00: Period = 0 SYSCLK cycles

0x01: Period = 2 SYSCLK cycles

0x02: Period = 4 SYSCLK cycles

...

0xFF: Period = 510 SYSCLK cycles

**Figure 8-22. GPBCTRL 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
QUALPRD3								QUALPRD2								QUALPRD1								QUALPRD0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 8-29. GPBCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	QUALPRD3	R/W	0h	Qualification sampling period for GPIO56 to GPIO63 Reset type: SYSRSn
23-16	QUALPRD2	R/W	0h	Qualification sampling period for GPIO48 to GPIO55 Reset type: SYSRSn
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO40 to GPIO47 Reset type: SYSRSn
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO32 to GPIO39 Reset type: SYSRSn



### 8.7.2.20 GPBQSEL1 Register (Offset = 42h) [reset = 0h]

GPBQSEL1 is shown in [Figure 8-23](#) and described in [Table 8-30](#).

Return to the [Summary Table](#).

GPIO B Qualification Type (GPIO32 to GPIO47)

Each field in this register selects the input qualification type for one IO pin. The available types are:

- 0: Synchronous
- 1: 3-sample qualification
- 2: 6-sample qualification
- 3: Asynchronous

**Figure 8-23. GPBQSEL1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO47		GPIO46		GPIO45		GPIO44		GPIO43		GPIO42		GPIO41		GPIO40	
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	7	6	5	4	3	2	1	0
GPIO39		RESERVED		GPIO37		RESERVED		GPIO35		GPIO34		GPIO33		GPIO32	
R/W-0h				R/W-0h				R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-30. GPBQSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO47	R/W	0h	Input qualification type for GPIO47 Reset type: SYSRSn
29-28	GPIO46	R/W	0h	Input qualification type for GPIO46 Reset type: SYSRSn
27-26	GPIO45	R/W	0h	Input qualification type for GPIO45 Reset type: SYSRSn
25-24	GPIO44	R/W	0h	Input qualification type for GPIO44 Reset type: SYSRSn
23-22	GPIO43	R/W	0h	Input qualification type for GPIO43 Reset type: SYSRSn
21-20	GPIO42	R/W	0h	Input qualification type for GPIO42 Reset type: SYSRSn
19-18	GPIO41	R/W	0h	Input qualification type for GPIO41 Reset type: SYSRSn
17-16	GPIO40	R/W	0h	Input qualification type for GPIO40 Reset type: SYSRSn
15-14	GPIO39	R/W	0h	Input qualification type for GPIO39 Reset type: SYSRSn
13-12	RESERVED	R/W	0h	Reserved
11-10	GPIO37	R/W	0h	Input qualification type for GPIO37 Reset type: SYSRSn
9-8	RESERVED	R/W	0h	Reserved
7-6	GPIO35	R/W	0h	Input qualification type for GPIO35 Reset type: SYSRSn
5-4	GPIO34	R/W	0h	Input qualification type for GPIO34 Reset type: SYSRSn
3-2	GPIO33	R/W	0h	Input qualification type for GPIO33 Reset type: SYSRSn

**Table 8-30. GPBQSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GPIO32	R/W	0h	Input qualification type for GPIO32 Reset type: SYSRSn

### 8.7.2.21 GPBQSEL2 Register (Offset = 44h) [reset = 0h]

GPBQSEL2 is shown in [Figure 8-24](#) and described in [Table 8-31](#).

Return to the [Summary Table](#).

GPIO B Qualification Type (GPIO48 to GPIO63)

Each field in this register determines the input qualification type for one IO pin. The available types are:

- 0: Synchronous
- 1: 3-sample qualification
- 2: 6-sample qualification
- 3: Asynchronous

**Figure 8-24. GPBQSEL2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
RESERVED				RESERVED				RESERVED				RESERVED				GPIO59		GPIO58		GPIO57		GPIO56	
																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								
GPIO55		GPIO54		GPIO53		GPIO52		GPIO51		GPIO50		GPIO49		GPIO48									
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 8-31. GPBQSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	GPIO59	R/W	0h	Input qualification type for GPIO59 Reset type: SYSRSn
21-20	GPIO58	R/W	0h	Input qualification type for GPIO58 Reset type: SYSRSn
19-18	GPIO57	R/W	0h	Input qualification type for GPIO57 Reset type: SYSRSn
17-16	GPIO56	R/W	0h	Input qualification type for GPIO56 Reset type: SYSRSn
15-14	GPIO55	R/W	0h	Input qualification type for GPIO55 Reset type: SYSRSn
13-12	GPIO54	R/W	0h	Input qualification type for GPIO54 Reset type: SYSRSn
11-10	GPIO53	R/W	0h	Input qualification type for GPIO53 Reset type: SYSRSn
9-8	GPIO52	R/W	0h	Input qualification type for GPIO52 Reset type: SYSRSn
7-6	GPIO51	R/W	0h	Input qualification type for GPIO51 Reset type: SYSRSn
5-4	GPIO50	R/W	0h	Input qualification type for GPIO50 Reset type: SYSRSn
3-2	GPIO49	R/W	0h	Input qualification type for GPIO49 Reset type: SYSRSn
1-0	GPIO48	R/W	0h	Input qualification type for GPIO48 Reset type: SYSRSn

### 8.7.2.22 GPBMUX1 Register (Offset = 46h) [reset = CC0h]

GPBMUX1 is shown in [Figure 8-25](#) and described in [Table 8-32](#).

Return to the [Summary Table](#).

GPIO B Peripheral Mux (GPIO32 to GPIO47)

Each field in this register determines part of the GPIO mux configuration for one IO pin. A value of 0x0, 0x4, 0x8, or 0xC configures the pin as a general-purpose IO. All other values select a peripheral to control the pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using this register before changing the corresponding field in the GPBGMUX1 register.

**Figure 8-25. GPBMUX1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO47		GPIO46		GPIO45		GPIO44		GPIO43		GPIO42		GPIO41		GPIO40	
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	7	6	5	4	3	2	1	0
GPIO39		RESERVED		GPIO37		RESERVED		GPIO35		GPIO34		GPIO33		GPIO32	
R/W-0h				R/W-3h				R/W-3h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-32. GPBMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO47	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO47 Reset type: SYSRSn
29-28	GPIO46	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO46 Reset type: SYSRSn
27-26	GPIO45	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO45 Reset type: SYSRSn
25-24	GPIO44	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO44 Reset type: SYSRSn
23-22	GPIO43	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO43 Reset type: SYSRSn
21-20	GPIO42	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO42 Reset type: SYSRSn
19-18	GPIO41	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO41 Reset type: SYSRSn
17-16	GPIO40	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO40 Reset type: SYSRSn
15-14	GPIO39	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO39 Reset type: SYSRSn
13-12	RESERVED	R/W	0h	Reserved
11-10	GPIO37	R/W	3h	Lower 2 bits of peripheral mux configuration for GPIO37 Reset type: SYSRSn
9-8	RESERVED	R/W	0h	Reserved
7-6	GPIO35	R/W	3h	Lower 2 bits of peripheral mux configuration for GPIO35 Reset type: SYSRSn
5-4	GPIO34	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO34 Reset type: SYSRSn
3-2	GPIO33	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO33 Reset type: SYSRSn
1-0	GPIO32	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO32 Reset type: SYSRSn

### 8.7.2.23 GPBMUX2 Register (Offset = 48h) [reset = 0h]

GPBMUX2 is shown in [Figure 8-26](#) and described in [Table 8-33](#).

Return to the [Summary Table](#).

GPIO B Peripheral Mux (GPIO48 to GPIO63)

Each field in this register determines part of the GPIO mux configuration for one IO pin. A value of 0x0, 0x4, 0x8, or 0xC configures the pin as a general-purpose IO. All other values select a peripheral to control the pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using this register before changing the corresponding field in the GPBGMUX2 register.

**Figure 8-26. GPBMUX2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				RESERVED				GPIO59		GPIO58		GPIO57		GPIO56	
								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
GPIO55		GPIO54		GPIO53		GPIO52		GPIO51		GPIO50		GPIO49		GPIO48	
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 8-33. GPBMUX2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	GPIO59	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO59 Reset type: SYSRSn
21-20	GPIO58	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO58 Reset type: SYSRSn
19-18	GPIO57	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO57 Reset type: SYSRSn
17-16	GPIO56	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO56 Reset type: SYSRSn
15-14	GPIO55	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO55 Reset type: SYSRSn
13-12	GPIO54	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO54 Reset type: SYSRSn
11-10	GPIO53	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO53 Reset type: SYSRSn
9-8	GPIO52	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO52 Reset type: SYSRSn
7-6	GPIO51	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO51 Reset type: SYSRSn
5-4	GPIO50	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO50 Reset type: SYSRSn
3-2	GPIO49	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO49 Reset type: SYSRSn
1-0	GPIO48	R/W	0h	Lower 2 bits of peripheral mux configuration for GPIO48 Reset type: SYSRSn

### 8.7.2.24 GPBDIR Register (Offset = 4Ah) [reset = 0h]

GPBDIR is shown in [Figure 8-27](#) and described in [Table 8-34](#).

Return to the [Summary Table](#).

GPIO B Direction (GPIO32 to GPIO63)

Each field in this register selects the direction of one IO pin in GPIO mode. If the pin is not configured as a general-purpose IO, this register has no effect.

0: The pin is an input

1: The pin is an output

**Figure 8-27. GPBDIR Register**

31		30		29		28		27		26		25		24	
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56	GPIO55	GPIO54	GPIO53	GPIO52
								R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48	GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40	GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32	GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h		R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-34. GPBDIR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	GPIO59	R/W	0h	Data direction for GPIO59 Reset type: SYSRSn
26	GPIO58	R/W	0h	Data direction for GPIO58 Reset type: SYSRSn
25	GPIO57	R/W	0h	Data direction for GPIO57 Reset type: SYSRSn
24	GPIO56	R/W	0h	Data direction for GPIO56 Reset type: SYSRSn
23	GPIO55	R/W	0h	Data direction for GPIO55 Reset type: SYSRSn
22	GPIO54	R/W	0h	Data direction for GPIO54 Reset type: SYSRSn
21	GPIO53	R/W	0h	Data direction for GPIO53 Reset type: SYSRSn
20	GPIO52	R/W	0h	Data direction for GPIO52 Reset type: SYSRSn
19	GPIO51	R/W	0h	Data direction for GPIO51 Reset type: SYSRSn
18	GPIO50	R/W	0h	Data direction for GPIO50 Reset type: SYSRSn

**Table 8-34. GPBDIR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO49	R/W	0h	Data direction for GPIO49 Reset type: SYSRSn
16	GPIO48	R/W	0h	Data direction for GPIO48 Reset type: SYSRSn
15	GPIO47	R/W	0h	Data direction for GPIO47 Reset type: SYSRSn
14	GPIO46	R/W	0h	Data direction for GPIO46 Reset type: SYSRSn
13	GPIO45	R/W	0h	Data direction for GPIO45 Reset type: SYSRSn
12	GPIO44	R/W	0h	Data direction for GPIO44 Reset type: SYSRSn
11	GPIO43	R/W	0h	Data direction for GPIO43 Reset type: SYSRSn
10	GPIO42	R/W	0h	Data direction for GPIO42 Reset type: SYSRSn
9	GPIO41	R/W	0h	Data direction for GPIO41 Reset type: SYSRSn
8	GPIO40	R/W	0h	Data direction for GPIO40 Reset type: SYSRSn
7	GPIO39	R/W	0h	Data direction for GPIO39 Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Data direction for GPIO37 Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Data direction for GPIO35 Reset type: SYSRSn
2	GPIO34	R/W	0h	Data direction for GPIO34 Reset type: SYSRSn
1	GPIO33	R/W	0h	Data direction for GPIO33 Reset type: SYSRSn
0	GPIO32	R/W	0h	Data direction for GPIO32 Reset type: SYSRSn

**8.7.2.25 GPBPUD Register (Offset = 4Ch) [reset = FFFFFFFFh]**

GPBPUD is shown in [Figure 8-28](#) and described in [Table 8-35](#).

Return to the [Summary Table](#).

GPIO B Pull-Up Disable (GPIO32 to GPIO63)

Each field in this register selects the state of the internal pull-up resistor for a single IO pin.

0: Pull-up enabled

1: Pull-up disabled

**Figure 8-28. GPBPUD Register**

31		30		29		28		27		26		25		24	
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56	GPIO55	GPIO54	GPIO53	GPIO52
								R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h
23		22		21		20		19		18		17		16	
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48	GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h
15		14		13		12		11		10		9		8	
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40	GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h		R/W-1h		R/W-1h	R/W-1h	R/W-1h	R/W-1h
7		6		5		4		3		2		1		0	
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32	GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-1h		R/W-1h		R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h

**Table 8-35. GPBPUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30	RESERVED	R/W	1h	Reserved
29	RESERVED	R/W	1h	Reserved
28	RESERVED	R/W	1h	Reserved
27	GPIO59	R/W	1h	Pull-up disable for GPIO59 Reset type: SYSRSn
26	GPIO58	R/W	1h	Pull-up disable for GPIO58 Reset type: SYSRSn
25	GPIO57	R/W	1h	Pull-up disable for GPIO57 Reset type: SYSRSn
24	GPIO56	R/W	1h	Pull-up disable for GPIO56 Reset type: SYSRSn
23	GPIO55	R/W	1h	Pull-up disable for GPIO55 Reset type: SYSRSn
22	GPIO54	R/W	1h	Pull-up disable for GPIO54 Reset type: SYSRSn
21	GPIO53	R/W	1h	Pull-up disable for GPIO53 Reset type: SYSRSn
20	GPIO52	R/W	1h	Pull-up disable for GPIO52 Reset type: SYSRSn
19	GPIO51	R/W	1h	Pull-up disable for GPIO51 Reset type: SYSRSn
18	GPIO50	R/W	1h	Pull-up disable for GPIO50 Reset type: SYSRSn



**Table 8-35. GPBPUD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO49	R/W	1h	Pull-up disable for GPIO49 Reset type: SYSRSn
16	GPIO48	R/W	1h	Pull-up disable for GPIO48 Reset type: SYSRSn
15	GPIO47	R/W	1h	Pull-up disable for GPIO47 Reset type: SYSRSn
14	GPIO46	R/W	1h	Pull-up disable for GPIO46 Reset type: SYSRSn
13	GPIO45	R/W	1h	Pull-up disable for GPIO45 Reset type: SYSRSn
12	GPIO44	R/W	1h	Pull-up disable for GPIO44 Reset type: SYSRSn
11	GPIO43	R/W	1h	Pull-up disable for GPIO43 Reset type: SYSRSn
10	GPIO42	R/W	1h	Pull-up disable for GPIO42 Reset type: SYSRSn
9	GPIO41	R/W	1h	Pull-up disable for GPIO41 Reset type: SYSRSn
8	GPIO40	R/W	1h	Pull-up disable for GPIO40 Reset type: SYSRSn
7	GPIO39	R/W	1h	Pull-up disable for GPIO39 Reset type: SYSRSn
6	RESERVED	R/W	1h	Reserved
5	GPIO37	R/W	1h	Pull-up disable for GPIO37 Reset type: SYSRSn
4	RESERVED	R/W	1h	Reserved
3	GPIO35	R/W	1h	Pull-up disable for GPIO35 Reset type: SYSRSn
2	GPIO34	R/W	1h	Pull-up disable for GPIO34 Reset type: SYSRSn
1	GPIO33	R/W	1h	Pull-up disable for GPIO33 Reset type: SYSRSn
0	GPIO32	R/W	1h	Pull-up disable for GPIO32 Reset type: SYSRSn

**8.7.2.26 GPBINV Register (Offset = 50h) [reset = 0h]**

GPBINV is shown in [Figure 8-29](#) and described in [Table 8-36](#).

Return to the [Summary Table](#).

GPIO B Input Inversion (GPIO32 to GPIO63)

Each field in this register selects whether the input value of one IO pin passes through an inverter.

0: The input is not inverted

1: The input is inverted

**Figure 8-29. GPBINV Register**

31		30		29		28		27		26		25		24	
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56	GPIO55	GPIO54	GPIO53	GPIO52
								R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48	GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40	GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32	GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h		R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-36. GPBINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	GPIO59	R/W	0h	Input inversion for GPIO59 Reset type: SYSRSn
26	GPIO58	R/W	0h	Input inversion for GPIO58 Reset type: SYSRSn
25	GPIO57	R/W	0h	Input inversion for GPIO57 Reset type: SYSRSn
24	GPIO56	R/W	0h	Input inversion for GPIO56 Reset type: SYSRSn
23	GPIO55	R/W	0h	Input inversion for GPIO55 Reset type: SYSRSn
22	GPIO54	R/W	0h	Input inversion for GPIO54 Reset type: SYSRSn
21	GPIO53	R/W	0h	Input inversion for GPIO53 Reset type: SYSRSn
20	GPIO52	R/W	0h	Input inversion for GPIO52 Reset type: SYSRSn
19	GPIO51	R/W	0h	Input inversion for GPIO51 Reset type: SYSRSn
18	GPIO50	R/W	0h	Input inversion for GPIO50 Reset type: SYSRSn

**Table 8-36. GPBINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO49	R/W	0h	Input inversion for GPIO49 Reset type: SYSRSn
16	GPIO48	R/W	0h	Input inversion for GPIO48 Reset type: SYSRSn
15	GPIO47	R/W	0h	Input inversion for GPIO47 Reset type: SYSRSn
14	GPIO46	R/W	0h	Input inversion for GPIO46 Reset type: SYSRSn
13	GPIO45	R/W	0h	Input inversion for GPIO45 Reset type: SYSRSn
12	GPIO44	R/W	0h	Input inversion for GPIO44 Reset type: SYSRSn
11	GPIO43	R/W	0h	Input inversion for GPIO43 Reset type: SYSRSn
10	GPIO42	R/W	0h	Input inversion for GPIO42 Reset type: SYSRSn
9	GPIO41	R/W	0h	Input inversion for GPIO41 Reset type: SYSRSn
8	GPIO40	R/W	0h	Input inversion for GPIO40 Reset type: SYSRSn
7	GPIO39	R/W	0h	Input inversion for GPIO39 Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Input inversion for GPIO37 Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Input inversion for GPIO35 Reset type: SYSRSn
2	GPIO34	R/W	0h	Input inversion for GPIO34 Reset type: SYSRSn
1	GPIO33	R/W	0h	Input inversion for GPIO33 Reset type: SYSRSn
0	GPIO32	R/W	0h	Input inversion for GPIO32 Reset type: SYSRSn

### 8.7.2.27 GPBODR Register (Offset = 52h) [reset = 0h]

GPBODR is shown in [Figure 8-30](#) and described in [Table 8-37](#).

Return to the [Summary Table](#).

GPIO B Open Drain Output Mode (GPIO32 to GPIO63)

Each field in this register selects between push-pull mode and open-drain mode for one general-purpose output pin. In both modes, writing a 0 to the output data latch drives the pin low. In push-pull mode, writing a 1 to the output data latch drives the pin high. In open-drain mode, it tri-states the output buffer.

0: Push-pull output

1: Open-drain output

**Figure 8-30. GPBODR Register**

31		30		29		28		27		26		25		24	
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56	GPIO55	GPIO54	GPIO53	GPIO52
								R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48	GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40	GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32	GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h		R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-37. GPBODR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	GPIO59	R/W	0h	Open-drain output mode for GPIO59 Reset type: SYSRSn
26	GPIO58	R/W	0h	Open-drain output mode for GPIO58 Reset type: SYSRSn
25	GPIO57	R/W	0h	Open-drain output mode for GPIO57 Reset type: SYSRSn
24	GPIO56	R/W	0h	Open-drain output mode for GPIO56 Reset type: SYSRSn
23	GPIO55	R/W	0h	Open-drain output mode for GPIO55 Reset type: SYSRSn
22	GPIO54	R/W	0h	Open-drain output mode for GPIO54 Reset type: SYSRSn
21	GPIO53	R/W	0h	Open-drain output mode for GPIO53 Reset type: SYSRSn
20	GPIO52	R/W	0h	Open-drain output mode for GPIO52 Reset type: SYSRSn
19	GPIO51	R/W	0h	Open-drain output mode for GPIO51 Reset type: SYSRSn

**Table 8-37. GPBODR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	GPIO50	R/W	0h	Open-drain output mode for GPIO50 Reset type: SYSRSn
17	GPIO49	R/W	0h	Open-drain output mode for GPIO49 Reset type: SYSRSn
16	GPIO48	R/W	0h	Open-drain output mode for GPIO48 Reset type: SYSRSn
15	GPIO47	R/W	0h	Open-drain output mode for GPIO47 Reset type: SYSRSn
14	GPIO46	R/W	0h	Open-drain output mode for GPIO46 Reset type: SYSRSn
13	GPIO45	R/W	0h	Open-drain output mode for GPIO45 Reset type: SYSRSn
12	GPIO44	R/W	0h	Open-drain output mode for GPIO44 Reset type: SYSRSn
11	GPIO43	R/W	0h	Open-drain output mode for GPIO43 Reset type: SYSRSn
10	GPIO42	R/W	0h	Open-drain output mode for GPIO42 Reset type: SYSRSn
9	GPIO41	R/W	0h	Open-drain output mode for GPIO41 Reset type: SYSRSn
8	GPIO40	R/W	0h	Open-drain output mode for GPIO40 Reset type: SYSRSn
7	GPIO39	R/W	0h	Open-drain output mode for GPIO39 Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Open-drain output mode for GPIO37 Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Open-drain output mode for GPIO35 Reset type: SYSRSn
2	GPIO34	R/W	0h	Open-drain output mode for GPIO34 Reset type: SYSRSn
1	GPIO33	R/W	0h	Open-drain output mode for GPIO33 Reset type: SYSRSn
0	GPIO32	R/W	0h	Open-drain output mode for GPIO32 Reset type: SYSRSn

### 8.7.2.28 GPBGMUX1 Register (Offset = 60h) [reset = CC0h]

GPBGMUX1 is shown in [Figure 8-31](#) and described in [Table 8-38](#).

Return to the [Summary Table](#).

GPIO B Peripheral Group Mux (GPIO32 to GPIO47)

Each field in this register determines part of the GPIO mux configuration for one IO pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using the GPBMUX1 register before changing their configuration in this register.

**Figure 8-31. GPBGMUX1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO47		GPIO46		GPIO45		GPIO44		GPIO43		GPIO42		GPIO41		GPIO40	
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	7	6	5	4	3	2	1	0
GPIO39		RESERVED		GPIO37		RESERVED		GPIO35		GPIO34		GPIO33		GPIO32	
R/W-0h				R/W-3h				R/W-3h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-38. GPBGMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO47	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO47 Reset type: SYSRSn
29-28	GPIO46	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO46 Reset type: SYSRSn
27-26	GPIO45	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO45 Reset type: SYSRSn
25-24	GPIO44	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO44 Reset type: SYSRSn
23-22	GPIO43	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO43 Reset type: SYSRSn
21-20	GPIO42	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO42 Reset type: SYSRSn
19-18	GPIO41	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO41 Reset type: SYSRSn
17-16	GPIO40	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO40 Reset type: SYSRSn
15-14	GPIO39	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO39 Reset type: SYSRSn
13-12	RESERVED	R/W	0h	Reserved
11-10	GPIO37	R/W	3h	Upper 2 bits of peripheral mux configuration for GPIO37 Reset type: SYSRSn
9-8	RESERVED	R/W	0h	Reserved
7-6	GPIO35	R/W	3h	Upper 2 bits of peripheral mux configuration for GPIO35 Reset type: SYSRSn
5-4	GPIO34	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO34 Reset type: SYSRSn
3-2	GPIO33	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO33 Reset type: SYSRSn
1-0	GPIO32	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO32 Reset type: SYSRSn

### 8.7.2.29 GPBGMUX2 Register (Offset = 62h) [reset = 0h]

GPBGMUX2 is shown in [Figure 8-32](#) and described in [Table 8-39](#).

Return to the [Summary Table](#).

GPIO B Peripheral Group Mux (GPIO48 to GPIO63)

Each field in this register determines part of the GPIO mux configuration for a single IO pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using the GPBMUX2 register before changing their configuration in this register.

**Figure 8-32. GPBGMUX2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
RESERVED				RESERVED				RESERVED				RESERVED				GPIO59		GPIO58		GPIO57		GPIO56	
																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								
GPIO55				GPIO54				GPIO53				GPIO52				GPIO51		GPIO50		GPIO49		GPIO48	
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 8-39. GPBGMUX2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	GPIO59	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO59 Reset type: SYSRSn
21-20	GPIO58	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO58 Reset type: SYSRSn
19-18	GPIO57	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO57 Reset type: SYSRSn
17-16	GPIO56	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO56 Reset type: SYSRSn
15-14	GPIO55	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO55 Reset type: SYSRSn
13-12	GPIO54	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO54 Reset type: SYSRSn
11-10	GPIO53	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO53 Reset type: SYSRSn
9-8	GPIO52	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO52 Reset type: SYSRSn
7-6	GPIO51	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO51 Reset type: SYSRSn
5-4	GPIO50	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO50 Reset type: SYSRSn
3-2	GPIO49	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO49 Reset type: SYSRSn
1-0	GPIO48	R/W	0h	Upper 2 bits of peripheral mux configuration for GPIO48 Reset type: SYSRSn

### 8.7.2.30 GPBCSEL1 Register (Offset = 68h) [reset = 0h]

GPBCSEL1 is shown in [Figure 8-33](#) and described in [Table 8-40](#).

Return to the [Summary Table](#).

GPIO B Master Core Select (GPIO32 to GPIO39)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPBDAT, GPBSET, GPBCLEAR, and GPBTOGGLE registers.

0x0: CPU is the master

0x1: CLA is the master

0x2 - 0xF: Reserved

**Figure 8-33. GPBCSEL1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO39				RESERVED				GPIO37				RESERVED			
R/W-0h								R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO35				GPIO34				GPIO33				GPIO32			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

**Table 8-40. GPBCSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	GPIO39	R/W	0h	Master core select for GPIO39 Reset type: SYSRSn
27-24	RESERVED	R/W	0h	Reserved
23-20	GPIO37	R/W	0h	Master core select for GPIO37 Reset type: SYSRSn
19-16	RESERVED	R/W	0h	Reserved
15-12	GPIO35	R/W	0h	Master core select for GPIO35 Reset type: SYSRSn
11-8	GPIO34	R/W	0h	Master core select for GPIO34 Reset type: SYSRSn
7-4	GPIO33	R/W	0h	Master core select for GPIO33 Reset type: SYSRSn
3-0	GPIO32	R/W	0h	Master core select for GPIO32 Reset type: SYSRSn



### 8.7.2.31 GPBCSEL2 Register (Offset = 6Ah) [reset = 0h]

GPBCSEL2 is shown in [Figure 8-34](#) and described in [Table 8-41](#).

Return to the [Summary Table](#).

GPIO B Master Core Select (GPIO40 to GPIO47)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPBDAT, GPBSET, GPBCLEAR, and GPBTOGGLE registers.

0x0: CPU is the master

0x1: CLA is the master

0x2 - 0xF: Reserved

**Figure 8-34. GPBCSEL2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO47				GPIO46				GPIO45				GPIO44			
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
GPIO43				GPIO42				GPIO41				GPIO40			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

**Table 8-41. GPBCSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	GPIO47	R/W	0h	Master core select for GPIO47 Reset type: SYSRSn
27-24	GPIO46	R/W	0h	Master core select for GPIO46 Reset type: SYSRSn
23-20	GPIO45	R/W	0h	Master core select for GPIO45 Reset type: SYSRSn
19-16	GPIO44	R/W	0h	Master core select for GPIO44 Reset type: SYSRSn
15-12	GPIO43	R/W	0h	Master core select for GPIO43 Reset type: SYSRSn
11-8	GPIO42	R/W	0h	Master core select for GPIO42 Reset type: SYSRSn
7-4	GPIO41	R/W	0h	Master core select for GPIO41 Reset type: SYSRSn
3-0	GPIO40	R/W	0h	Master core select for GPIO40 Reset type: SYSRSn

**8.7.2.32 GPBCSEL3 Register (Offset = 6Ch) [reset = 0h]**

GPBCSEL3 is shown in [Figure 8-35](#) and described in [Table 8-42](#).

Return to the [Summary Table](#).

GPIO B Master Core Select (GPIO48 to GPIO55)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPBDAT, GPBSET, GPBCLEAR, and GPBTOGGLE registers.

0x0: CPU is the master

0x1: CLA is the master

0x2 - 0xF: Reserved

**Figure 8-35. GPBCSEL3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO55				GPIO54				GPIO53				GPIO52			
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
GPIO51				GPIO50				GPIO49				GPIO48			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

**Table 8-42. GPBCSEL3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	GPIO55	R/W	0h	Master core select for GPIO55 Reset type: SYSRSn
27-24	GPIO54	R/W	0h	Master core select for GPIO54 Reset type: SYSRSn
23-20	GPIO53	R/W	0h	Master core select for GPIO53 Reset type: SYSRSn
19-16	GPIO52	R/W	0h	Master core select for GPIO52 Reset type: SYSRSn
15-12	GPIO51	R/W	0h	Master core select for GPIO51 Reset type: SYSRSn
11-8	GPIO50	R/W	0h	Master core select for GPIO50 Reset type: SYSRSn
7-4	GPIO49	R/W	0h	Master core select for GPIO49 Reset type: SYSRSn
3-0	GPIO48	R/W	0h	Master core select for GPIO48 Reset type: SYSRSn

### 8.7.2.33 GPBCSEL4 Register (Offset = 6Eh) [reset = 0h]

GPBCSEL4 is shown in [Figure 8-36](#) and described in [Table 8-43](#).

Return to the [Summary Table](#).

GPIO B Master Core Select (GPIO56 to GPIO63)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPBDAT, GPBSET, GPBCLEAR, and GPBTOGGLE registers.

0x0: CPU is the master

0x1: CLA is the master

0x2 - 0xF: Reserved

**Figure 8-36. GPBCSEL4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				RESERVED				RESERVED				RESERVED			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO59				GPIO58				GPIO57				GPIO56			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

**Table 8-43. GPBCSEL4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R/W	0h	Reserved
27-24	RESERVED	R/W	0h	Reserved
23-20	RESERVED	R/W	0h	Reserved
19-16	RESERVED	R/W	0h	Reserved
15-12	GPIO59	R/W	0h	Master core select for GPIO59 Reset type: SYSRSn
11-8	GPIO58	R/W	0h	Master core select for GPIO58 Reset type: SYSRSn
7-4	GPIO57	R/W	0h	Master core select for GPIO57 Reset type: SYSRSn
3-0	GPIO56	R/W	0h	Master core select for GPIO56 Reset type: SYSRSn

### 8.7.2.34 GPBLOCK Register (Offset = 7Ch) [reset = 0h]

GPBLOCK is shown in [Figure 8-37](#) and described in [Table 8-44](#).

Return to the [Summary Table](#).

GPIO B Lock Register (GPIO32 to GPIO63)

Each field in this register locks one IO pin's configuration. This blocks writes to the corresponding bits in the GPBMUXn, GPBIDR, GPBINV, GPBODR, GPBGMUXn, and GPBCSELn registers.

0: Pin configuration is unlocked

1: Pin configuration is locked

**Figure 8-37. GPBLOCK Register**

31		30		29		28		27		26		25		24	
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56	GPIO55	GPIO54	GPIO53	GPIO52
								R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48	GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40	GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32	GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h		R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-44. GPBLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	GPIO59	R/W	0h	Configuration lock for GPIO59 Reset type: SYSRSn
26	GPIO58	R/W	0h	Configuration lock for GPIO58 Reset type: SYSRSn
25	GPIO57	R/W	0h	Configuration lock for GPIO57 Reset type: SYSRSn
24	GPIO56	R/W	0h	Configuration lock for GPIO56 Reset type: SYSRSn
23	GPIO55	R/W	0h	Configuration lock for GPIO55 Reset type: SYSRSn
22	GPIO54	R/W	0h	Configuration lock for GPIO54 Reset type: SYSRSn
21	GPIO53	R/W	0h	Configuration lock for GPIO53 Reset type: SYSRSn
20	GPIO52	R/W	0h	Configuration lock for GPIO52 Reset type: SYSRSn
19	GPIO51	R/W	0h	Configuration lock for GPIO51 Reset type: SYSRSn
18	GPIO50	R/W	0h	Configuration lock for GPIO50 Reset type: SYSRSn

**Table 8-44. GPBLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO49	R/W	0h	Configuration lock for GPIO49 Reset type: SYSRSn
16	GPIO48	R/W	0h	Configuration lock for GPIO48 Reset type: SYSRSn
15	GPIO47	R/W	0h	Configuration lock for GPIO47 Reset type: SYSRSn
14	GPIO46	R/W	0h	Configuration lock for GPIO46 Reset type: SYSRSn
13	GPIO45	R/W	0h	Configuration lock for GPIO45 Reset type: SYSRSn
12	GPIO44	R/W	0h	Configuration lock for GPIO44 Reset type: SYSRSn
11	GPIO43	R/W	0h	Configuration lock for GPIO43 Reset type: SYSRSn
10	GPIO42	R/W	0h	Configuration lock for GPIO42 Reset type: SYSRSn
9	GPIO41	R/W	0h	Configuration lock for GPIO41 Reset type: SYSRSn
8	GPIO40	R/W	0h	Configuration lock for GPIO40 Reset type: SYSRSn
7	GPIO39	R/W	0h	Configuration lock for GPIO39 Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Configuration lock for GPIO37 Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Configuration lock for GPIO35 Reset type: SYSRSn
2	GPIO34	R/W	0h	Configuration lock for GPIO34 Reset type: SYSRSn
1	GPIO33	R/W	0h	Configuration lock for GPIO33 Reset type: SYSRSn
0	GPIO32	R/W	0h	Configuration lock for GPIO32 Reset type: SYSRSn

### 8.7.2.35 GPBCR Register (Offset = 7Eh) [reset = 0h]

GPBCR is shown in [Figure 8-38](#) and described in [Table 8-45](#).

Return to the [Summary Table](#).

GPIO B Lock Commit Register (GPIO32 to GPIO63)

Each field in this register blocks writes to one IO pin's GPBLOCK bit. Once set, a lock commit can only be cleared by a reset.

0: Pin configuration lock is unlocked

1: Pin configuration lock is locked

**Figure 8-38. GPBCR Register**

31		30		29		28		27		26		25		24	
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56	GPIO55	GPIO54	GPIO53	GPIO52
								R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	
23		22		21		20		19		18		17		16	
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48	GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h
15		14		13		12		11		10		9		8	
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40	GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h
7		6		5		4		3		2		1		0	
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32	GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/WOnce-0h		R/WOnce-0h		R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 8-45. GPBCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/WOnce	0h	Reserved
30	RESERVED	R/WOnce	0h	Reserved
29	RESERVED	R/WOnce	0h	Reserved
28	RESERVED	R/WOnce	0h	Reserved
27	GPIO59	R/WOnce	0h	Configuration lock commit for GPIO59 Reset type: SYSRSn
26	GPIO58	R/WOnce	0h	Configuration lock commit for GPIO58 Reset type: SYSRSn
25	GPIO57	R/WOnce	0h	Configuration lock commit for GPIO57 Reset type: SYSRSn
24	GPIO56	R/WOnce	0h	Configuration lock commit for GPIO56 Reset type: SYSRSn
23	GPIO55	R/WOnce	0h	Configuration lock commit for GPIO55 Reset type: SYSRSn
22	GPIO54	R/WOnce	0h	Configuration lock commit for GPIO54 Reset type: SYSRSn
21	GPIO53	R/WOnce	0h	Configuration lock commit for GPIO53 Reset type: SYSRSn
20	GPIO52	R/WOnce	0h	Configuration lock commit for GPIO52 Reset type: SYSRSn
19	GPIO51	R/WOnce	0h	Configuration lock commit for GPIO51 Reset type: SYSRSn
18	GPIO50	R/WOnce	0h	Configuration lock commit for GPIO50 Reset type: SYSRSn

**Table 8-45. GPBCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO49	R/WOnce	0h	Configuration lock commit for GPIO49 Reset type: SYSRSn
16	GPIO48	R/WOnce	0h	Configuration lock commit for GPIO48 Reset type: SYSRSn
15	GPIO47	R/WOnce	0h	Configuration lock commit for GPIO47 Reset type: SYSRSn
14	GPIO46	R/WOnce	0h	Configuration lock commit for GPIO46 Reset type: SYSRSn
13	GPIO45	R/WOnce	0h	Configuration lock commit for GPIO45 Reset type: SYSRSn
12	GPIO44	R/WOnce	0h	Configuration lock commit for GPIO44 Reset type: SYSRSn
11	GPIO43	R/WOnce	0h	Configuration lock commit for GPIO43 Reset type: SYSRSn
10	GPIO42	R/WOnce	0h	Configuration lock commit for GPIO42 Reset type: SYSRSn
9	GPIO41	R/WOnce	0h	Configuration lock commit for GPIO41 Reset type: SYSRSn
8	GPIO40	R/WOnce	0h	Configuration lock commit for GPIO40 Reset type: SYSRSn
7	GPIO39	R/WOnce	0h	Configuration lock commit for GPIO39 Reset type: SYSRSn
6	RESERVED	R/WOnce	0h	Reserved
5	GPIO37	R/WOnce	0h	Configuration lock commit for GPIO37 Reset type: SYSRSn
4	RESERVED	R/WOnce	0h	Reserved
3	GPIO35	R/WOnce	0h	Configuration lock commit for GPIO35 Reset type: SYSRSn
2	GPIO34	R/WOnce	0h	Configuration lock commit for GPIO34 Reset type: SYSRSn
1	GPIO33	R/WOnce	0h	Configuration lock commit for GPIO33 Reset type: SYSRSn
0	GPIO32	R/WOnce	0h	Configuration lock commit for GPIO32 Reset type: SYSRSn

### 8.7.2.36 GPHCTRL Register (Offset = 1C0h) [reset = 0h]

GPHCTRL is shown in [Figure 8-39](#) and described in [Table 8-46](#).

Return to the [Summary Table](#).

GPIO H Qualification Sampling Period (GPIO224 to GPIO255)

Each field in this register selects the qualification sampling period in SYSCLK cycles for eight GPIOs. The period is equal to 2 times the register field value.

0x00: Period = 0 SYSCLK cycles

0x01: Period = 2 SYSCLK cycles

0x02: Period = 4 SYSCLK cycles

...

0xFF: Period = 510 SYSCLK cycles

**Figure 8-39. GPHCTRL 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								QUALPRD2								QUALPRD1								QUALPRD0							
								R/W-0h								R/W-0h								R/W-0h							

**Table 8-46. GPHCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R/W	0h	Reserved
23-16	QUALPRD2	R/W	0h	Qualification sampling period for GPIO240 to GPIO247 Reset type: SYSRSn
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO232 to GPIO239 Reset type: SYSRSn
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO224 to GPIO231 Reset type: SYSRSn



### 8.7.2.37 GPHQSEL1 Register (Offset = 1C2h) [reset = 0h]

GPHQSEL1 is shown in [Figure 8-40](#) and described in [Table 8-47](#).

Return to the [Summary Table](#).

GPIO H Qualification Type (GPIO224 to GPIO239)

Each field in this register selects the input qualification type for one IO pin. The available types are:

- 0: Synchronous
- 1: 3-sample qualification
- 2: 6-sample qualification
- 3: Asynchronous

**Figure 8-40. GPHQSEL1 Register**

31	30	29	28	27	26	25	24
GPIO239	GPIO238		GPIO237		GPIO236		
R/W-0h	R/W-0h		R/W-0h		R/W-0h		
23	22	21	20	19	18	17	16
GPIO235	GPIO234		GPIO233		GPIO232		
R/W-0h	R/W-0h		R/W-0h		R/W-0h		
15	14	13	12	11	10	9	8
GPIO231	GPIO230		GPIO229		GPIO228		
R/W-0h	R/W-0h		R/W-0h		R/W-0h		
7	6	5	4	3	2	1	0
GPIO227	GPIO226		GPIO225		GPIO224		
R/W-0h	R/W-0h		R/W-0h		R/W-0h		

**Table 8-47. GPHQSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO239	R/W	0h	Input qualification type for GPIO239 Reset type: SYSRSn
29-28	GPIO238	R/W	0h	Input qualification type for GPIO238 Reset type: SYSRSn
27-26	GPIO237	R/W	0h	Input qualification type for GPIO237 Reset type: SYSRSn
25-24	GPIO236	R/W	0h	Input qualification type for GPIO236 Reset type: SYSRSn
23-22	GPIO235	R/W	0h	Input qualification type for GPIO235 Reset type: SYSRSn
21-20	GPIO234	R/W	0h	Input qualification type for GPIO234 Reset type: SYSRSn
19-18	GPIO233	R/W	0h	Input qualification type for GPIO233 Reset type: SYSRSn
17-16	GPIO232	R/W	0h	Input qualification type for GPIO232 Reset type: SYSRSn
15-14	GPIO231	R/W	0h	Input qualification type for GPIO231 Reset type: SYSRSn
13-12	GPIO230	R/W	0h	Input qualification type for GPIO230 Reset type: SYSRSn
11-10	GPIO229	R/W	0h	Input qualification type for GPIO229 Reset type: SYSRSn

**Table 8-47. GPHQSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	GPIO228	R/W	0h	Input qualification type for GPIO228 Reset type: SYSRSn
7-6	GPIO227	R/W	0h	Input qualification type for GPIO227 Reset type: SYSRSn
5-4	GPIO226	R/W	0h	Input qualification type for GPIO226 Reset type: SYSRSn
3-2	GPIO225	R/W	0h	Input qualification type for GPIO225 Reset type: SYSRSn
1-0	GPIO224	R/W	0h	Input qualification type for GPIO224 Reset type: SYSRSn

**8.7.2.38 GPHQSEL2 Register (Offset = 1C4h) [reset = 0h]**

GPHQSEL2 is shown in [Figure 8-41](#) and described in [Table 8-48](#).

Return to the [Summary Table](#).

GPIO H Qualification Type (GPIO240 to GPIO255)

Each field in this register determines the input qualification type for one IO pin. The available types are:

- 0: Synchronous
- 1: 3-sample qualification
- 2: 6-sample qualification
- 3: Asynchronous

**Figure 8-41. GPHQSEL2 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED		RESERVED		RESERVED	
23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED	
15	14	13	12	11	10	9	8
GPIO247		GPIO246		GPIO245		GPIO244	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO243		GPIO242		GPIO241		GPIO240	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-48. GPHQSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15-14	GPIO247	R/W	0h	Input qualification type for GPIO247 Reset type: SYSRSn
13-12	GPIO246	R/W	0h	Input qualification type for GPIO246 Reset type: SYSRSn
11-10	GPIO245	R/W	0h	Input qualification type for GPIO245 Reset type: SYSRSn
9-8	GPIO244	R/W	0h	Input qualification type for GPIO244 Reset type: SYSRSn
7-6	GPIO243	R/W	0h	Input qualification type for GPIO243 Reset type: SYSRSn
5-4	GPIO242	R/W	0h	Input qualification type for GPIO242 Reset type: SYSRSn
3-2	GPIO241	R/W	0h	Input qualification type for GPIO241 Reset type: SYSRSn

**Table 8-48. GPHQSEL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GPIO240	R/W	0h	Input qualification type for GPIO240 Reset type: SYSRSn

### 8.7.2.39 GPHPUD Register (Offset = 1CCh) [reset = FFFFFFFFh]

GPHPUD is shown in [Figure 8-42](#) and described in [Table 8-49](#).

Return to the [Summary Table](#).

GPIO H Pull-Up Disable (GPIO224 to GPIO255)

Each field in this register selects the state of the internal pull-up resistor for a single IO pin.

0: Pull-up enabled

1: Pull-up disabled

**Figure 8-42. GPHPUD Register**

31		30		29		28		27		26		25		24	
RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED	
23		22		21		20		19		18		17		16	
GPIO247		GPIO246		GPIO245		GPIO244		GPIO243		GPIO242		GPIO241		GPIO240	
R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h	
15		14		13		12		11		10		9		8	
GPIO239		GPIO238		GPIO237		GPIO236		GPIO235		GPIO234		GPIO233		GPIO232	
R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h	
7		6		5		4		3		2		1		0	
GPIO231		GPIO230		GPIO229		GPIO228		GPIO227		GPIO226		GPIO225		GPIO224	
R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h	

**Table 8-49. GPHPUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30	RESERVED	R/W	1h	Reserved
29	RESERVED	R/W	1h	Reserved
28	RESERVED	R/W	1h	Reserved
27	RESERVED	R/W	1h	Reserved
26	RESERVED	R/W	1h	Reserved
25	RESERVED	R/W	1h	Reserved
24	RESERVED	R/W	1h	Reserved
23	GPIO247	R/W	1h	Pull-up disable for GPIO247 Reset type: SYSRSn
22	GPIO246	R/W	1h	Pull-up disable for GPIO246 Reset type: SYSRSn
21	GPIO245	R/W	1h	Pull-up disable for GPIO245 Reset type: SYSRSn
20	GPIO244	R/W	1h	Pull-up disable for GPIO244 Reset type: SYSRSn
19	GPIO243	R/W	1h	Pull-up disable for GPIO243 Reset type: SYSRSn
18	GPIO242	R/W	1h	Pull-up disable for GPIO242 Reset type: SYSRSn
17	GPIO241	R/W	1h	Pull-up disable for GPIO241 Reset type: SYSRSn
16	GPIO240	R/W	1h	Pull-up disable for GPIO240 Reset type: SYSRSn

**Table 8-49. GPHPUD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	GPIO239	R/W	1h	Pull-up disable for GPIO239 Reset type: SYSRSn
14	GPIO238	R/W	1h	Pull-up disable for GPIO238 Reset type: SYSRSn
13	GPIO237	R/W	1h	Pull-up disable for GPIO237 Reset type: SYSRSn
12	GPIO236	R/W	1h	Pull-up disable for GPIO236 Reset type: SYSRSn
11	GPIO235	R/W	1h	Pull-up disable for GPIO235 Reset type: SYSRSn
10	GPIO234	R/W	1h	Pull-up disable for GPIO234 Reset type: SYSRSn
9	GPIO233	R/W	1h	Pull-up disable for GPIO233 Reset type: SYSRSn
8	GPIO232	R/W	1h	Pull-up disable for GPIO232 Reset type: SYSRSn
7	GPIO231	R/W	1h	Pull-up disable for GPIO231 Reset type: SYSRSn
6	GPIO230	R/W	1h	Pull-up disable for GPIO230 Reset type: SYSRSn
5	GPIO229	R/W	1h	Pull-up disable for GPIO229 Reset type: SYSRSn
4	GPIO228	R/W	1h	Pull-up disable for GPIO228 Reset type: SYSRSn
3	GPIO227	R/W	1h	Pull-up disable for GPIO227 Reset type: SYSRSn
2	GPIO226	R/W	1h	Pull-up disable for GPIO226 Reset type: SYSRSn
1	GPIO225	R/W	1h	Pull-up disable for GPIO225 Reset type: SYSRSn
0	GPIO224	R/W	1h	Pull-up disable for GPIO224 Reset type: SYSRSn

### 8.7.2.40 GPHINV Register (Offset = 1D0h) [reset = 0h]

GPHINV is shown in [Figure 8-43](#) and described in [Table 8-50](#).

Return to the [Summary Table](#).

GPIO H Input Inversion (GPIO224 to GPIO255)

Each field in this register selects whether the input value of one IO pin passes through an inverter.

0: The input is not inverted

1: The input is inverted

**Figure 8-43. GPHINV Register**

31		30		29		28		27		26		25		24	
RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED	
23		22		21		20		19		18		17		16	
GPIO247		GPIO246		GPIO245		GPIO244		GPIO243		GPIO242		GPIO241		GPIO240	
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	
GPIO239		GPIO238		GPIO237		GPIO236		GPIO235		GPIO234		GPIO233		GPIO232	
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	
GPIO231		GPIO230		GPIO229		GPIO228		GPIO227		GPIO226		GPIO225		GPIO224	
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 8-50. GPHINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	GPIO247	R/W	0h	Input inversion for GPIO247 Reset type: SYSRSn
22	GPIO246	R/W	0h	Input inversion for GPIO246 Reset type: SYSRSn
21	GPIO245	R/W	0h	Input inversion for GPIO245 Reset type: SYSRSn
20	GPIO244	R/W	0h	Input inversion for GPIO244 Reset type: SYSRSn
19	GPIO243	R/W	0h	Input inversion for GPIO243 Reset type: SYSRSn
18	GPIO242	R/W	0h	Input inversion for GPIO242 Reset type: SYSRSn
17	GPIO241	R/W	0h	Input inversion for GPIO241 Reset type: SYSRSn
16	GPIO240	R/W	0h	Input inversion for GPIO240 Reset type: SYSRSn

**Table 8-50. GPHINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	GPIO239	R/W	0h	Input inversion for GPIO239 Reset type: SYSRSn
14	GPIO238	R/W	0h	Input inversion for GPIO238 Reset type: SYSRSn
13	GPIO237	R/W	0h	Input inversion for GPIO237 Reset type: SYSRSn
12	GPIO236	R/W	0h	Input inversion for GPIO236 Reset type: SYSRSn
11	GPIO235	R/W	0h	Input inversion for GPIO235 Reset type: SYSRSn
10	GPIO234	R/W	0h	Input inversion for GPIO234 Reset type: SYSRSn
9	GPIO233	R/W	0h	Input inversion for GPIO233 Reset type: SYSRSn
8	GPIO232	R/W	0h	Input inversion for GPIO232 Reset type: SYSRSn
7	GPIO231	R/W	0h	Input inversion for GPIO231 Reset type: SYSRSn
6	GPIO230	R/W	0h	Input inversion for GPIO230 Reset type: SYSRSn
5	GPIO229	R/W	0h	Input inversion for GPIO229 Reset type: SYSRSn
4	GPIO228	R/W	0h	Input inversion for GPIO228 Reset type: SYSRSn
3	GPIO227	R/W	0h	Input inversion for GPIO227 Reset type: SYSRSn
2	GPIO226	R/W	0h	Input inversion for GPIO226 Reset type: SYSRSn
1	GPIO225	R/W	0h	Input inversion for GPIO225 Reset type: SYSRSn
0	GPIO224	R/W	0h	Input inversion for GPIO224 Reset type: SYSRSn



**8.7.2.41 GPHAMSEL Register (Offset = 1D4h) [reset = FFFFFFFFh]**

GPHAMSEL is shown in [Figure 8-44](#) and described in [Table 8-51](#).

Return to the [Summary Table](#).

GPIO H Analog Mode Select (GPIO224 to GPIO255)

Each field in this register selects between analog and digital functionality for one IO pin.

- 0: Digital mode
- 1: Analog mode

**Figure 8-44. GPHAMSEL Register**

31		30		29		28		27		26		25		24	
RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED	
23		22		21		20		19		18		17		16	
GPIO247		GPIO246		GPIO245		GPIO244		GPIO243		GPIO242		GPIO241		GPIO240	
R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h	
15		14		13		12		11		10		9		8	
GPIO239		GPIO238		GPIO237		GPIO236		GPIO235		GPIO234		GPIO233		GPIO232	
R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h	
7		6		5		4		3		2		1		0	
GPIO231		GPIO230		GPIO229		GPIO228		GPIO227		GPIO226		GPIO225		GPIO224	
R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h		R/W-1h	

**Table 8-51. GPHAMSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30	RESERVED	R/W	1h	Reserved
29	RESERVED	R/W	1h	Reserved
28	RESERVED	R/W	1h	Reserved
27	RESERVED	R/W	1h	Reserved
26	RESERVED	R/W	1h	Reserved
25	RESERVED	R/W	1h	Reserved
24	RESERVED	R/W	1h	Reserved
23	GPIO247	R/W	1h	Analog mode select for GPIO247 Reset type: SYSRSn
22	GPIO246	R/W	1h	Analog mode select for GPIO246 Reset type: SYSRSn
21	GPIO245	R/W	1h	Analog mode select for GPIO245 Reset type: SYSRSn
20	GPIO244	R/W	1h	Analog mode select for GPIO244 Reset type: SYSRSn
19	GPIO243	R/W	1h	Analog mode select for GPIO243 Reset type: SYSRSn
18	GPIO242	R/W	1h	Analog mode select for GPIO242 Reset type: SYSRSn
17	GPIO241	R/W	1h	Analog mode select for GPIO241 Reset type: SYSRSn
16	GPIO240	R/W	1h	Analog mode select for GPIO240 Reset type: SYSRSn

**Table 8-51. GPHAMSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	GPIO239	R/W	1h	Analog mode select for GPIO239 Reset type: SYSRSn
14	GPIO238	R/W	1h	Analog mode select for GPIO238 Reset type: SYSRSn
13	GPIO237	R/W	1h	Analog mode select for GPIO237 Reset type: SYSRSn
12	GPIO236	R/W	1h	Analog mode select for GPIO236 Reset type: SYSRSn
11	GPIO235	R/W	1h	Analog mode select for GPIO235 Reset type: SYSRSn
10	GPIO234	R/W	1h	Analog mode select for GPIO234 Reset type: SYSRSn
9	GPIO233	R/W	1h	Analog mode select for GPIO233 Reset type: SYSRSn
8	GPIO232	R/W	1h	Analog mode select for GPIO232 Reset type: SYSRSn
7	GPIO231	R/W	1h	Analog mode select for GPIO231 Reset type: SYSRSn
6	GPIO230	R/W	1h	Analog mode select for GPIO230 Reset type: SYSRSn
5	GPIO229	R/W	1h	Analog mode select for GPIO229 Reset type: SYSRSn
4	GPIO228	R/W	1h	Analog mode select for GPIO228 Reset type: SYSRSn
3	GPIO227	R/W	1h	Analog mode select for GPIO227 Reset type: SYSRSn
2	GPIO226	R/W	1h	Analog mode select for GPIO226 Reset type: SYSRSn
1	GPIO225	R/W	1h	Analog mode select for GPIO225 Reset type: SYSRSn
0	GPIO224	R/W	1h	Analog mode select for GPIO224 Reset type: SYSRSn

### 8.7.2.42 GPHLOCK Register (Offset = 1FCh) [reset = 0h]

GPHLOCK is shown in [Figure 8-45](#) and described in [Table 8-52](#).

Return to the [Summary Table](#).

GPIO H Lock Register (GPIO224 to GPIO255)

Each field in this register locks one IO pin's configuration. This blocks writes to the corresponding bits in the GPHINV register.

0: Pin configuration is unlocked

1: Pin configuration is locked

**Figure 8-45. GPHLOCK Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
23	22	21	20	19	18	17	16
GPIO247	GPIO246	GPIO245	GPIO244	GPIO243	GPIO242	GPIO241	GPIO240
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
GPIO239	GPIO238	GPIO237	GPIO236	GPIO235	GPIO234	GPIO233	GPIO232
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
GPIO231	GPIO230	GPIO229	GPIO228	GPIO227	GPIO226	GPIO225	GPIO224
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 8-52. GPHLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	GPIO247	R/W	0h	Configuration lock for GPIO247 Reset type: SYSRSn
22	GPIO246	R/W	0h	Configuration lock for GPIO246 Reset type: SYSRSn
21	GPIO245	R/W	0h	Configuration lock for GPIO245 Reset type: SYSRSn
20	GPIO244	R/W	0h	Configuration lock for GPIO244 Reset type: SYSRSn
19	GPIO243	R/W	0h	Configuration lock for GPIO243 Reset type: SYSRSn
18	GPIO242	R/W	0h	Configuration lock for GPIO242 Reset type: SYSRSn
17	GPIO241	R/W	0h	Configuration lock for GPIO241 Reset type: SYSRSn
16	GPIO240	R/W	0h	Configuration lock for GPIO240 Reset type: SYSRSn

**Table 8-52. GPHLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	GPIO239	R/W	0h	Configuration lock for GPIO239 Reset type: SYSRSn
14	GPIO238	R/W	0h	Configuration lock for GPIO238 Reset type: SYSRSn
13	GPIO237	R/W	0h	Configuration lock for GPIO237 Reset type: SYSRSn
12	GPIO236	R/W	0h	Configuration lock for GPIO236 Reset type: SYSRSn
11	GPIO235	R/W	0h	Configuration lock for GPIO235 Reset type: SYSRSn
10	GPIO234	R/W	0h	Configuration lock for GPIO234 Reset type: SYSRSn
9	GPIO233	R/W	0h	Configuration lock for GPIO233 Reset type: SYSRSn
8	GPIO232	R/W	0h	Configuration lock for GPIO232 Reset type: SYSRSn
7	GPIO231	R/W	0h	Configuration lock for GPIO231 Reset type: SYSRSn
6	GPIO230	R/W	0h	Configuration lock for GPIO230 Reset type: SYSRSn
5	GPIO229	R/W	0h	Configuration lock for GPIO229 Reset type: SYSRSn
4	GPIO228	R/W	0h	Configuration lock for GPIO228 Reset type: SYSRSn
3	GPIO227	R/W	0h	Configuration lock for GPIO227 Reset type: SYSRSn
2	GPIO226	R/W	0h	Configuration lock for GPIO226 Reset type: SYSRSn
1	GPIO225	R/W	0h	Configuration lock for GPIO225 Reset type: SYSRSn
0	GPIO224	R/W	0h	Configuration lock for GPIO224 Reset type: SYSRSn

### 8.7.2.43 GPHCR Register (Offset = 1FEh) [reset = 0h]

GPHCR is shown in [Figure 8-46](#) and described in [Table 8-53](#).

Return to the [Summary Table](#).

GPIO H Lock Commit Register (GPIO224 to GPIO255)

Each field in this register blocks writes to one IO pin's GPHLOCK bit. Once set, a lock commit can only be cleared by a reset.

0: Pin configuration lock is unlocked

1: Pin configuration lock is locked

**Figure 8-46. GPHCR Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
23	22	21	20	19	18	17	16
GPIO247	GPIO246	GPIO245	GPIO244	GPIO243	GPIO242	GPIO241	GPIO240
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h
15	14	13	12	11	10	9	8
GPIO239	GPIO238	GPIO237	GPIO236	GPIO235	GPIO234	GPIO233	GPIO232
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h
7	6	5	4	3	2	1	0
GPIO231	GPIO230	GPIO229	GPIO228	GPIO227	GPIO226	GPIO225	GPIO224
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 8-53. GPHCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/WOnce	0h	Reserved
30	RESERVED	R/WOnce	0h	Reserved
29	RESERVED	R/WOnce	0h	Reserved
28	RESERVED	R/WOnce	0h	Reserved
27	RESERVED	R/WOnce	0h	Reserved
26	RESERVED	R/WOnce	0h	Reserved
25	RESERVED	R/WOnce	0h	Reserved
24	RESERVED	R/WOnce	0h	Reserved
23	GPIO247	R/WOnce	0h	Configuration lock commit for GPIO247 Reset type: SYSRSn
22	GPIO246	R/WOnce	0h	Configuration lock commit for GPIO246 Reset type: SYSRSn
21	GPIO245	R/WOnce	0h	Configuration lock commit for GPIO245 Reset type: SYSRSn
20	GPIO244	R/WOnce	0h	Configuration lock commit for GPIO244 Reset type: SYSRSn
19	GPIO243	R/WOnce	0h	Configuration lock commit for GPIO243 Reset type: SYSRSn
18	GPIO242	R/WOnce	0h	Configuration lock commit for GPIO242 Reset type: SYSRSn
17	GPIO241	R/WOnce	0h	Configuration lock commit for GPIO241 Reset type: SYSRSn
16	GPIO240	R/WOnce	0h	Configuration lock commit for GPIO240 Reset type: SYSRSn

**Table 8-53. GPHCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	GPIO239	R/WOnce	0h	Configuration lock commit for GPIO239 Reset type: SYSRSn
14	GPIO238	R/WOnce	0h	Configuration lock commit for GPIO238 Reset type: SYSRSn
13	GPIO237	R/WOnce	0h	Configuration lock commit for GPIO237 Reset type: SYSRSn
12	GPIO236	R/WOnce	0h	Configuration lock commit for GPIO236 Reset type: SYSRSn
11	GPIO235	R/WOnce	0h	Configuration lock commit for GPIO235 Reset type: SYSRSn
10	GPIO234	R/WOnce	0h	Configuration lock commit for GPIO234 Reset type: SYSRSn
9	GPIO233	R/WOnce	0h	Configuration lock commit for GPIO233 Reset type: SYSRSn
8	GPIO232	R/WOnce	0h	Configuration lock commit for GPIO232 Reset type: SYSRSn
7	GPIO231	R/WOnce	0h	Configuration lock commit for GPIO231 Reset type: SYSRSn
6	GPIO230	R/WOnce	0h	Configuration lock commit for GPIO230 Reset type: SYSRSn
5	GPIO229	R/WOnce	0h	Configuration lock commit for GPIO229 Reset type: SYSRSn
4	GPIO228	R/WOnce	0h	Configuration lock commit for GPIO228 Reset type: SYSRSn
3	GPIO227	R/WOnce	0h	Configuration lock commit for GPIO227 Reset type: SYSRSn
2	GPIO226	R/WOnce	0h	Configuration lock commit for GPIO226 Reset type: SYSRSn
1	GPIO225	R/WOnce	0h	Configuration lock commit for GPIO225 Reset type: SYSRSn
0	GPIO224	R/WOnce	0h	Configuration lock commit for GPIO224 Reset type: SYSRSn

### 8.7.3 GPIO\_DATA\_REGS Registers

Table 8-54 lists the GPIO\_DATA\_REGS registers. All register offset addresses not listed in Table 8-54 should be considered as reserved locations and the register contents should not be modified.

**Table 8-54. GPIO\_DATA\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	GPADAT	GPIO A Data Register (GPIO0 to GPIO31)		<a href="#">Go</a>
2h	GPASET	GPIO A Output Set (GPIO0 to GPIO31)		<a href="#">Go</a>
4h	GPACLEAR	GPIO A Output Clear (GPIO0 to GPIO31)		<a href="#">Go</a>
6h	GPATOGGLE	GPIO A Output Toggle (GPIO0 to GPIO31)		<a href="#">Go</a>
8h	GPBDAT	GPIO B Data Register (GPIO32 to GPIO64)		<a href="#">Go</a>
Ah	GPBSET	GPIO B Output Set (GPIO32 to GPIO64)		<a href="#">Go</a>
Ch	GPBCLEAR	GPIO B Output Clear (GPIO32 to GPIO64)		<a href="#">Go</a>
Eh	GPBTOGGLE	GPIO B Output Toggle (GPIO32 to GPIO64)		<a href="#">Go</a>
38h	GPHDAT	GPIO H Data Register (GPIO0 to GPIO255)		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 8-55 shows the codes that are used for access types in this section.

**Table 8-55. GPIO\_DATA\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 8.7.3.1 GPADAT Register (Offset = 0h) [reset = 0h]

GPADAT is shown in [Figure 8-47](#) and described in [Table 8-56](#).

Return to the [Summary Table](#).

GPIO A Data (GPIO0 to GPIO31)

Reading a field in this register returns the input level of the corresponding IO pin after qualification and (optionally) inversion. The input value is always readable even when the pin is configured as a GPIO output or peripheral signal.

Writing to a field in this register selects the value of the output data latch for the corresponding IO pin. If the pin is configured as a GPIO output, this value will be driven onto the pin. Otherwise, the value will be latched and ignored unless the pin is reconfigured to be an output. A system reset will clear all output latches.

Due to the difference between the read and write values, sequential read-modify-write operations on this register may corrupt the state of the output latches. The GPASET, GPACLEAR, and GPATOGGLE registers should be used to safely control the output latches.

**Figure 8-47. GPADAT Register**

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
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
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
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
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
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 8-56. GPADAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Read: Input value, Write: Output latch for GPIO31 Reset type: SYSRSn
30	GPIO30	R/W	0h	Read: Input value, Write: Output latch for GPIO30 Reset type: SYSRSn
29	GPIO29	R/W	0h	Read: Input value, Write: Output latch for GPIO29 Reset type: SYSRSn
28	GPIO28	R/W	0h	Read: Input value, Write: Output latch for GPIO28 Reset type: SYSRSn
27	GPIO27	R/W	0h	Read: Input value, Write: Output latch for GPIO27 Reset type: SYSRSn
26	GPIO26	R/W	0h	Read: Input value, Write: Output latch for GPIO26 Reset type: SYSRSn
25	GPIO25	R/W	0h	Read: Input value, Write: Output latch for GPIO25 Reset type: SYSRSn
24	GPIO24	R/W	0h	Read: Input value, Write: Output latch for GPIO24 Reset type: SYSRSn
23	GPIO23	R/W	0h	Read: Input value, Write: Output latch for GPIO23 Reset type: SYSRSn



**Table 8-56. GPADAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	GPIO22	R/W	0h	Read: Input value, Write: Output latch for GPIO22 Reset type: SYSRSn
21	GPIO21	R/W	0h	Read: Input value, Write: Output latch for GPIO21 Reset type: SYSRSn
20	GPIO20	R/W	0h	Read: Input value, Write: Output latch for GPIO20 Reset type: SYSRSn
19	GPIO19	R/W	0h	Read: Input value, Write: Output latch for GPIO19 Reset type: SYSRSn
18	GPIO18	R/W	0h	Read: Input value, Write: Output latch for GPIO18 Reset type: SYSRSn
17	GPIO17	R/W	0h	Read: Input value, Write: Output latch for GPIO17 Reset type: SYSRSn
16	GPIO16	R/W	0h	Read: Input value, Write: Output latch for GPIO16 Reset type: SYSRSn
15	GPIO15	R/W	0h	Read: Input value, Write: Output latch for GPIO15 Reset type: SYSRSn
14	GPIO14	R/W	0h	Read: Input value, Write: Output latch for GPIO14 Reset type: SYSRSn
13	GPIO13	R/W	0h	Read: Input value, Write: Output latch for GPIO13 Reset type: SYSRSn
12	GPIO12	R/W	0h	Read: Input value, Write: Output latch for GPIO12 Reset type: SYSRSn
11	GPIO11	R/W	0h	Read: Input value, Write: Output latch for GPIO11 Reset type: SYSRSn
10	GPIO10	R/W	0h	Read: Input value, Write: Output latch for GPIO10 Reset type: SYSRSn
9	GPIO9	R/W	0h	Read: Input value, Write: Output latch for GPIO9 Reset type: SYSRSn
8	GPIO8	R/W	0h	Read: Input value, Write: Output latch for GPIO8 Reset type: SYSRSn
7	GPIO7	R/W	0h	Read: Input value, Write: Output latch for GPIO7 Reset type: SYSRSn
6	GPIO6	R/W	0h	Read: Input value, Write: Output latch for GPIO6 Reset type: SYSRSn
5	GPIO5	R/W	0h	Read: Input value, Write: Output latch for GPIO5 Reset type: SYSRSn
4	GPIO4	R/W	0h	Read: Input value, Write: Output latch for GPIO4 Reset type: SYSRSn
3	GPIO3	R/W	0h	Read: Input value, Write: Output latch for GPIO3 Reset type: SYSRSn
2	GPIO2	R/W	0h	Read: Input value, Write: Output latch for GPIO2 Reset type: SYSRSn
1	GPIO1	R/W	0h	Read: Input value, Write: Output latch for GPIO1 Reset type: SYSRSn
0	GPIO0	R/W	0h	Read: Input value, Write: Output latch for GPIO0 Reset type: SYSRSn

### 8.7.3.2 GPASET Register (Offset = 2h) [reset = 0h]

GPASET is shown in [Figure 8-48](#) and described in [Table 8-57](#).

Return to the [Summary Table](#).

GPIO Output Set (GPIO0 to GPIO31)

Writing a 1 to a field in this register sets the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-48. GPASET Register**

31		30		29		28		27		26		25		24	
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24	GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16	GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-57. GPASET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Output set for GPIO31 Reset type: SYSRSn
30	GPIO30	R/W	0h	Output set for GPIO30 Reset type: SYSRSn
29	GPIO29	R/W	0h	Output set for GPIO29 Reset type: SYSRSn
28	GPIO28	R/W	0h	Output set for GPIO28 Reset type: SYSRSn
27	GPIO27	R/W	0h	Output set for GPIO27 Reset type: SYSRSn
26	GPIO26	R/W	0h	Output set for GPIO26 Reset type: SYSRSn
25	GPIO25	R/W	0h	Output set for GPIO25 Reset type: SYSRSn
24	GPIO24	R/W	0h	Output set for GPIO24 Reset type: SYSRSn
23	GPIO23	R/W	0h	Output set for GPIO23 Reset type: SYSRSn
22	GPIO22	R/W	0h	Output set for GPIO22 Reset type: SYSRSn
21	GPIO21	R/W	0h	Output set for GPIO21 Reset type: SYSRSn
20	GPIO20	R/W	0h	Output set for GPIO20 Reset type: SYSRSn
19	GPIO19	R/W	0h	Output set for GPIO19 Reset type: SYSRSn

**Table 8-57. GPASET Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	GPIO18	R/W	0h	Output set for GPIO18 Reset type: SYSRSn
17	GPIO17	R/W	0h	Output set for GPIO17 Reset type: SYSRSn
16	GPIO16	R/W	0h	Output set for GPIO16 Reset type: SYSRSn
15	GPIO15	R/W	0h	Output set for GPIO15 Reset type: SYSRSn
14	GPIO14	R/W	0h	Output set for GPIO14 Reset type: SYSRSn
13	GPIO13	R/W	0h	Output set for GPIO13 Reset type: SYSRSn
12	GPIO12	R/W	0h	Output set for GPIO12 Reset type: SYSRSn
11	GPIO11	R/W	0h	Output set for GPIO11 Reset type: SYSRSn
10	GPIO10	R/W	0h	Output set for GPIO10 Reset type: SYSRSn
9	GPIO9	R/W	0h	Output set for GPIO9 Reset type: SYSRSn
8	GPIO8	R/W	0h	Output set for GPIO8 Reset type: SYSRSn
7	GPIO7	R/W	0h	Output set for GPIO7 Reset type: SYSRSn
6	GPIO6	R/W	0h	Output set for GPIO6 Reset type: SYSRSn
5	GPIO5	R/W	0h	Output set for GPIO5 Reset type: SYSRSn
4	GPIO4	R/W	0h	Output set for GPIO4 Reset type: SYSRSn
3	GPIO3	R/W	0h	Output set for GPIO3 Reset type: SYSRSn
2	GPIO2	R/W	0h	Output set for GPIO2 Reset type: SYSRSn
1	GPIO1	R/W	0h	Output set for GPIO1 Reset type: SYSRSn
0	GPIO0	R/W	0h	Output set for GPIO0 Reset type: SYSRSn

### 8.7.3.3 GPACLEAR Register (Offset = 4h) [reset = 0h]

GPACLEAR is shown in [Figure 8-49](#) and described in [Table 8-58](#).

Return to the [Summary Table](#).

GPIO Output Clear (GPIO0 to GPIO31)

Writing a 1 to a field in this register clears the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-49. GPACLEAR Register**

31		30		29		28		27		26		25		24	
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24	GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16	GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-58. GPACLEAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Output clear for GPIO31 Reset type: SYSRSn
30	GPIO30	R/W	0h	Output clear for GPIO30 Reset type: SYSRSn
29	GPIO29	R/W	0h	Output clear for GPIO29 Reset type: SYSRSn
28	GPIO28	R/W	0h	Output clear for GPIO28 Reset type: SYSRSn
27	GPIO27	R/W	0h	Output clear for GPIO27 Reset type: SYSRSn
26	GPIO26	R/W	0h	Output clear for GPIO26 Reset type: SYSRSn
25	GPIO25	R/W	0h	Output clear for GPIO25 Reset type: SYSRSn
24	GPIO24	R/W	0h	Output clear for GPIO24 Reset type: SYSRSn
23	GPIO23	R/W	0h	Output clear for GPIO23 Reset type: SYSRSn
22	GPIO22	R/W	0h	Output clear for GPIO22 Reset type: SYSRSn
21	GPIO21	R/W	0h	Output clear for GPIO21 Reset type: SYSRSn
20	GPIO20	R/W	0h	Output clear for GPIO20 Reset type: SYSRSn
19	GPIO19	R/W	0h	Output clear for GPIO19 Reset type: SYSRSn

**Table 8-58. GPACLEAR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	GPIO18	R/W	0h	Output clear for GPIO18 Reset type: SYSRSn
17	GPIO17	R/W	0h	Output clear for GPIO17 Reset type: SYSRSn
16	GPIO16	R/W	0h	Output clear for GPIO16 Reset type: SYSRSn
15	GPIO15	R/W	0h	Output clear for GPIO15 Reset type: SYSRSn
14	GPIO14	R/W	0h	Output clear for GPIO14 Reset type: SYSRSn
13	GPIO13	R/W	0h	Output clear for GPIO13 Reset type: SYSRSn
12	GPIO12	R/W	0h	Output clear for GPIO12 Reset type: SYSRSn
11	GPIO11	R/W	0h	Output clear for GPIO11 Reset type: SYSRSn
10	GPIO10	R/W	0h	Output clear for GPIO10 Reset type: SYSRSn
9	GPIO9	R/W	0h	Output clear for GPIO9 Reset type: SYSRSn
8	GPIO8	R/W	0h	Output clear for GPIO8 Reset type: SYSRSn
7	GPIO7	R/W	0h	Output clear for GPIO7 Reset type: SYSRSn
6	GPIO6	R/W	0h	Output clear for GPIO6 Reset type: SYSRSn
5	GPIO5	R/W	0h	Output clear for GPIO5 Reset type: SYSRSn
4	GPIO4	R/W	0h	Output clear for GPIO4 Reset type: SYSRSn
3	GPIO3	R/W	0h	Output clear for GPIO3 Reset type: SYSRSn
2	GPIO2	R/W	0h	Output clear for GPIO2 Reset type: SYSRSn
1	GPIO1	R/W	0h	Output clear for GPIO1 Reset type: SYSRSn
0	GPIO0	R/W	0h	Output clear for GPIO0 Reset type: SYSRSn

### 8.7.3.4 GPATOGGLE Register (Offset = 6h) [reset = 0h]

GPATOGGLE is shown in [Figure 8-50](#) and described in [Table 8-59](#).

Return to the [Summary Table](#).

GPIO Output Toggle (GPIO0 to GPIO31)

Writing a 1 to a field in this register inverts the value of the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-50. GPATOGGLE Register**

31		30		29		28		27		26		25		24	
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24	GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16	GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-59. GPATOGGLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Output toggle for GPIO31 Reset type: SYSRSn
30	GPIO30	R/W	0h	Output toggle for GPIO30 Reset type: SYSRSn
29	GPIO29	R/W	0h	Output toggle for GPIO29 Reset type: SYSRSn
28	GPIO28	R/W	0h	Output toggle for GPIO28 Reset type: SYSRSn
27	GPIO27	R/W	0h	Output toggle for GPIO27 Reset type: SYSRSn
26	GPIO26	R/W	0h	Output toggle for GPIO26 Reset type: SYSRSn
25	GPIO25	R/W	0h	Output toggle for GPIO25 Reset type: SYSRSn
24	GPIO24	R/W	0h	Output toggle for GPIO24 Reset type: SYSRSn
23	GPIO23	R/W	0h	Output toggle for GPIO23 Reset type: SYSRSn
22	GPIO22	R/W	0h	Output toggle for GPIO22 Reset type: SYSRSn
21	GPIO21	R/W	0h	Output toggle for GPIO21 Reset type: SYSRSn
20	GPIO20	R/W	0h	Output toggle for GPIO20 Reset type: SYSRSn
19	GPIO19	R/W	0h	Output toggle for GPIO19 Reset type: SYSRSn

**Table 8-59. GPATOGGLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	GPIO18	R/W	0h	Output toggle for GPIO18 Reset type: SYSRSn
17	GPIO17	R/W	0h	Output toggle for GPIO17 Reset type: SYSRSn
16	GPIO16	R/W	0h	Output toggle for GPIO16 Reset type: SYSRSn
15	GPIO15	R/W	0h	Output toggle for GPIO15 Reset type: SYSRSn
14	GPIO14	R/W	0h	Output toggle for GPIO14 Reset type: SYSRSn
13	GPIO13	R/W	0h	Output toggle for GPIO13 Reset type: SYSRSn
12	GPIO12	R/W	0h	Output toggle for GPIO12 Reset type: SYSRSn
11	GPIO11	R/W	0h	Output toggle for GPIO11 Reset type: SYSRSn
10	GPIO10	R/W	0h	Output toggle for GPIO10 Reset type: SYSRSn
9	GPIO9	R/W	0h	Output toggle for GPIO9 Reset type: SYSRSn
8	GPIO8	R/W	0h	Output toggle for GPIO8 Reset type: SYSRSn
7	GPIO7	R/W	0h	Output toggle for GPIO7 Reset type: SYSRSn
6	GPIO6	R/W	0h	Output toggle for GPIO6 Reset type: SYSRSn
5	GPIO5	R/W	0h	Output toggle for GPIO5 Reset type: SYSRSn
4	GPIO4	R/W	0h	Output toggle for GPIO4 Reset type: SYSRSn
3	GPIO3	R/W	0h	Output toggle for GPIO3 Reset type: SYSRSn
2	GPIO2	R/W	0h	Output toggle for GPIO2 Reset type: SYSRSn
1	GPIO1	R/W	0h	Output toggle for GPIO1 Reset type: SYSRSn
0	GPIO0	R/W	0h	Output toggle for GPIO0 Reset type: SYSRSn

### 8.7.3.5 GPBDAT Register (Offset = 8h) [reset = 0h]

GPBDAT is shown in [Figure 8-51](#) and described in [Table 8-60](#).

Return to the [Summary Table](#).

GPIO B Data (GPIO32 to GPIO64)

Reading a field in this register returns the input level of the corresponding IO pin after qualification and (optionally) inversion. The input value is always readable even when the pin is configured as a GPIO output or peripheral signal.

Writing to a field in this register selects the value of the output data latch for the corresponding IO pin. If the pin is configured as a GPIO output, this value will be driven onto the pin. Otherwise, the value will be latched and ignored unless the pin is reconfigured to be an output. A system reset will clear all output latches.

Due to the difference between the read and write values, sequential read-modify-write operations on this register may corrupt the state of the output latches. The GPBSET, GPBCLEAR, and GPBTOGGLE registers should be used to safely control the output latches.

**Figure 8-51. GPBDAT Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56
				R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48
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
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
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
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0h		R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 8-60. GPBDAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	GPIO59	R/W	0h	Read: Input value, Write: Output latch for GPIO59 Reset type: SYSRSn
26	GPIO58	R/W	0h	Read: Input value, Write: Output latch for GPIO58 Reset type: SYSRSn
25	GPIO57	R/W	0h	Read: Input value, Write: Output latch for GPIO57 Reset type: SYSRSn
24	GPIO56	R/W	0h	Read: Input value, Write: Output latch for GPIO56 Reset type: SYSRSn
23	GPIO55	R/W	0h	Read: Input value, Write: Output latch for GPIO55 Reset type: SYSRSn
22	GPIO54	R/W	0h	Read: Input value, Write: Output latch for GPIO54 Reset type: SYSRSn
21	GPIO53	R/W	0h	Read: Input value, Write: Output latch for GPIO53 Reset type: SYSRSn



**Table 8-60. GPBDAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	GPIO52	R/W	0h	Read: Input value, Write: Output latch for GPIO52 Reset type: SYSRSn
19	GPIO51	R/W	0h	Read: Input value, Write: Output latch for GPIO51 Reset type: SYSRSn
18	GPIO50	R/W	0h	Read: Input value, Write: Output latch for GPIO50 Reset type: SYSRSn
17	GPIO49	R/W	0h	Read: Input value, Write: Output latch for GPIO49 Reset type: SYSRSn
16	GPIO48	R/W	0h	Read: Input value, Write: Output latch for GPIO48 Reset type: SYSRSn
15	GPIO47	R/W	0h	Read: Input value, Write: Output latch for GPIO47 Reset type: SYSRSn
14	GPIO46	R/W	0h	Read: Input value, Write: Output latch for GPIO46 Reset type: SYSRSn
13	GPIO45	R/W	0h	Read: Input value, Write: Output latch for GPIO45 Reset type: SYSRSn
12	GPIO44	R/W	0h	Read: Input value, Write: Output latch for GPIO44 Reset type: SYSRSn
11	GPIO43	R/W	0h	Read: Input value, Write: Output latch for GPIO43 Reset type: SYSRSn
10	GPIO42	R/W	0h	Read: Input value, Write: Output latch for GPIO42 Reset type: SYSRSn
9	GPIO41	R/W	0h	Read: Input value, Write: Output latch for GPIO41 Reset type: SYSRSn
8	GPIO40	R/W	0h	Read: Input value, Write: Output latch for GPIO40 Reset type: SYSRSn
7	GPIO39	R/W	0h	Read: Input value, Write: Output latch for GPIO39 Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Read: Input value, Write: Output latch for GPIO37 Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Read: Input value, Write: Output latch for GPIO35 Reset type: SYSRSn
2	GPIO34	R/W	0h	Read: Input value, Write: Output latch for GPIO34 Reset type: SYSRSn
1	GPIO33	R/W	0h	Read: Input value, Write: Output latch for GPIO33 Reset type: SYSRSn
0	GPIO32	R/W	0h	Read: Input value, Write: Output latch for GPIO32 Reset type: SYSRSn

### 8.7.3.6 GPBSET Register (Offset = Ah) [reset = 0h]

GPBSET is shown in [Figure 8-52](#) and described in [Table 8-61](#).

Return to the [Summary Table](#).

GPIO Output Set (GPIO32 to GPIO64)

Writing a 1 to a field in this register sets the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-52. GPBSET Register**

31		30		29		28		27		26		25		24	
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56	GPIO55	GPIO54	GPIO53	GPIO52
								R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48	GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40	GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32	GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h		R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-61. GPBSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	GPIO59	R/W	0h	Output set for GPIO59 Reset type: SYSRSn
26	GPIO58	R/W	0h	Output set for GPIO58 Reset type: SYSRSn
25	GPIO57	R/W	0h	Output set for GPIO57 Reset type: SYSRSn
24	GPIO56	R/W	0h	Output set for GPIO56 Reset type: SYSRSn
23	GPIO55	R/W	0h	Output set for GPIO55 Reset type: SYSRSn
22	GPIO54	R/W	0h	Output set for GPIO54 Reset type: SYSRSn
21	GPIO53	R/W	0h	Output set for GPIO53 Reset type: SYSRSn
20	GPIO52	R/W	0h	Output set for GPIO52 Reset type: SYSRSn
19	GPIO51	R/W	0h	Output set for GPIO51 Reset type: SYSRSn
18	GPIO50	R/W	0h	Output set for GPIO50 Reset type: SYSRSn

**Table 8-61. GPBSET Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO49	R/W	0h	Output set for GPIO49 Reset type: SYSRSn
16	GPIO48	R/W	0h	Output set for GPIO48 Reset type: SYSRSn
15	GPIO47	R/W	0h	Output set for GPIO47 Reset type: SYSRSn
14	GPIO46	R/W	0h	Output set for GPIO46 Reset type: SYSRSn
13	GPIO45	R/W	0h	Output set for GPIO45 Reset type: SYSRSn
12	GPIO44	R/W	0h	Output set for GPIO44 Reset type: SYSRSn
11	GPIO43	R/W	0h	Output set for GPIO43 Reset type: SYSRSn
10	GPIO42	R/W	0h	Output set for GPIO42 Reset type: SYSRSn
9	GPIO41	R/W	0h	Output set for GPIO41 Reset type: SYSRSn
8	GPIO40	R/W	0h	Output set for GPIO40 Reset type: SYSRSn
7	GPIO39	R/W	0h	Output set for GPIO39 Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Output set for GPIO37 Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Output set for GPIO35 Reset type: SYSRSn
2	GPIO34	R/W	0h	Output set for GPIO34 Reset type: SYSRSn
1	GPIO33	R/W	0h	Output set for GPIO33 Reset type: SYSRSn
0	GPIO32	R/W	0h	Output set for GPIO32 Reset type: SYSRSn

### 8.7.3.7 GPBCLEAR Register (Offset = Ch) [reset = 0h]

GPBCLEAR is shown in [Figure 8-53](#) and described in [Table 8-62](#).

Return to the [Summary Table](#).

GPIO Output Clear (GPIO32 to GPIO64)

Writing a 1 to a field in this register clears the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-53. GPBCLEAR Register**

31		30		29		28		27		26		25		24	
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56	GPIO55	GPIO54	GPIO53	GPIO52
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23		22		21		20		19		18		17		16	
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48	GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
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	
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40	GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
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	
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32	GPIO31	RESERVED	GPIO29	RESERVED	GPIO27	RESERVED	GPIO25	RESERVED
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 8-62. GPBCLEAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	GPIO59	R/W	0h	Output clear for GPIO59 Reset type: SYSRSn
26	GPIO58	R/W	0h	Output clear for GPIO58 Reset type: SYSRSn
25	GPIO57	R/W	0h	Output clear for GPIO57 Reset type: SYSRSn
24	GPIO56	R/W	0h	Output clear for GPIO56 Reset type: SYSRSn
23	GPIO55	R/W	0h	Output clear for GPIO55 Reset type: SYSRSn
22	GPIO54	R/W	0h	Output clear for GPIO54 Reset type: SYSRSn
21	GPIO53	R/W	0h	Output clear for GPIO53 Reset type: SYSRSn
20	GPIO52	R/W	0h	Output clear for GPIO52 Reset type: SYSRSn
19	GPIO51	R/W	0h	Output clear for GPIO51 Reset type: SYSRSn
18	GPIO50	R/W	0h	Output clear for GPIO50 Reset type: SYSRSn

**Table 8-62. GPBCLEAR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO49	R/W	0h	Output clear for GPIO49 Reset type: SYSRSn
16	GPIO48	R/W	0h	Output clear for GPIO48 Reset type: SYSRSn
15	GPIO47	R/W	0h	Output clear for GPIO47 Reset type: SYSRSn
14	GPIO46	R/W	0h	Output clear for GPIO46 Reset type: SYSRSn
13	GPIO45	R/W	0h	Output clear for GPIO45 Reset type: SYSRSn
12	GPIO44	R/W	0h	Output clear for GPIO44 Reset type: SYSRSn
11	GPIO43	R/W	0h	Output clear for GPIO43 Reset type: SYSRSn
10	GPIO42	R/W	0h	Output clear for GPIO42 Reset type: SYSRSn
9	GPIO41	R/W	0h	Output clear for GPIO41 Reset type: SYSRSn
8	GPIO40	R/W	0h	Output clear for GPIO40 Reset type: SYSRSn
7	GPIO39	R/W	0h	Output clear for GPIO39 Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Output clear for GPIO37 Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Output clear for GPIO35 Reset type: SYSRSn
2	GPIO34	R/W	0h	Output clear for GPIO34 Reset type: SYSRSn
1	GPIO33	R/W	0h	Output clear for GPIO33 Reset type: SYSRSn
0	GPIO32	R/W	0h	Output clear for GPIO32 Reset type: SYSRSn

### 8.7.3.8 GPBTOGGLE Register (Offset = Eh) [reset = 0h]

GPBTOGGLE is shown in [Figure 8-54](#) and described in [Table 8-63](#).

Return to the [Summary Table](#).

GPIO Output Toggle (GPIO32 to GPIO64)

Writing a 1 to a field in this register inverts the value of the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-54. GPBTOGGLE Register**

31		30		29		28		27		26		25		24	
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO59	GPIO58	GPIO57	GPIO56	GPIO55	GPIO54	GPIO53	GPIO52
								R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48	GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40	GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	GPIO34	GPIO33	GPIO32	GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h		R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 8-63. GPBTOGGLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	GPIO59	R/W	0h	Output toggle for GPIO59 Reset type: SYSRSn
26	GPIO58	R/W	0h	Output toggle for GPIO58 Reset type: SYSRSn
25	GPIO57	R/W	0h	Output toggle for GPIO57 Reset type: SYSRSn
24	GPIO56	R/W	0h	Output toggle for GPIO56 Reset type: SYSRSn
23	GPIO55	R/W	0h	Output toggle for GPIO55 Reset type: SYSRSn
22	GPIO54	R/W	0h	Output toggle for GPIO54 Reset type: SYSRSn
21	GPIO53	R/W	0h	Output toggle for GPIO53 Reset type: SYSRSn
20	GPIO52	R/W	0h	Output toggle for GPIO52 Reset type: SYSRSn
19	GPIO51	R/W	0h	Output toggle for GPIO51 Reset type: SYSRSn
18	GPIO50	R/W	0h	Output toggle for GPIO50 Reset type: SYSRSn

**Table 8-63. GPBTOGGLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO49	R/W	0h	Output toggle for GPIO49 Reset type: SYSRSn
16	GPIO48	R/W	0h	Output toggle for GPIO48 Reset type: SYSRSn
15	GPIO47	R/W	0h	Output toggle for GPIO47 Reset type: SYSRSn
14	GPIO46	R/W	0h	Output toggle for GPIO46 Reset type: SYSRSn
13	GPIO45	R/W	0h	Output toggle for GPIO45 Reset type: SYSRSn
12	GPIO44	R/W	0h	Output toggle for GPIO44 Reset type: SYSRSn
11	GPIO43	R/W	0h	Output toggle for GPIO43 Reset type: SYSRSn
10	GPIO42	R/W	0h	Output toggle for GPIO42 Reset type: SYSRSn
9	GPIO41	R/W	0h	Output toggle for GPIO41 Reset type: SYSRSn
8	GPIO40	R/W	0h	Output toggle for GPIO40 Reset type: SYSRSn
7	GPIO39	R/W	0h	Output toggle for GPIO39 Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Output toggle for GPIO37 Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Output toggle for GPIO35 Reset type: SYSRSn
2	GPIO34	R/W	0h	Output toggle for GPIO34 Reset type: SYSRSn
1	GPIO33	R/W	0h	Output toggle for GPIO33 Reset type: SYSRSn
0	GPIO32	R/W	0h	Output toggle for GPIO32 Reset type: SYSRSn

### 8.7.3.9 GPHDAT Register (Offset = 38h) [reset = 0h]

GPHDAT is shown in [Figure 8-55](#) and described in [Table 8-64](#).

Return to the [Summary Table](#).

GPIO H Data (GPIO224 to GPIO255)

Reading a field in this register returns the input level of the corresponding IO pin after qualification and (optionally) inversion. In digital mode, these pins only support input functionality.

**Figure 8-55. GPHDAT Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
23	22	21	20	19	18	17	16
GPIO247	GPIO246	GPIO245	GPIO244	GPIO243	GPIO242	GPIO241	GPIO240
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
GPIO239	GPIO238	GPIO237	GPIO236	GPIO235	GPIO234	GPIO233	GPIO232
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
GPIO231	GPIO230	GPIO229	GPIO228	GPIO227	GPIO226	GPIO225	GPIO224
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 8-64. GPHDAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	GPIO247	R/W	0h	Read: Input value for GPIO247 Reset type: SYSRSn
22	GPIO246	R/W	0h	Read: Input value for GPIO246 Reset type: SYSRSn
21	GPIO245	R/W	0h	Read: Input value for GPIO245 Reset type: SYSRSn
20	GPIO244	R/W	0h	Read: Input value for GPIO244 Reset type: SYSRSn
19	GPIO243	R/W	0h	Read: Input value for GPIO243 Reset type: SYSRSn
18	GPIO242	R/W	0h	Read: Input value for GPIO242 Reset type: SYSRSn
17	GPIO241	R/W	0h	Read: Input value for GPIO241 Reset type: SYSRSn
16	GPIO240	R/W	0h	Read: Input value for GPIO240 Reset type: SYSRSn



**Table 8-64. GPHDAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	GPIO239	R/W	0h	Read: Input value for GPIO239 Reset type: SYSRSn
14	GPIO238	R/W	0h	Read: Input value for GPIO238 Reset type: SYSRSn
13	GPIO237	R/W	0h	Read: Input value for GPIO237 Reset type: SYSRSn
12	GPIO236	R/W	0h	Read: Input value for GPIO236 Reset type: SYSRSn
11	GPIO235	R/W	0h	Read: Input value for GPIO235 Reset type: SYSRSn
10	GPIO234	R/W	0h	Read: Input value for GPIO234 Reset type: SYSRSn
9	GPIO233	R/W	0h	Read: Input value for GPIO233 Reset type: SYSRSn
8	GPIO232	R/W	0h	Read: Input value for GPIO232 Reset type: SYSRSn
7	GPIO231	R/W	0h	Read: Input value for GPIO231 Reset type: SYSRSn
6	GPIO230	R/W	0h	Read: Input value for GPIO230 Reset type: SYSRSn
5	GPIO229	R/W	0h	Read: Input value for GPIO229 Reset type: SYSRSn
4	GPIO228	R/W	0h	Read: Input value for GPIO228 Reset type: SYSRSn
3	GPIO227	R/W	0h	Read: Input value for GPIO227 Reset type: SYSRSn
2	GPIO226	R/W	0h	Read: Input value for GPIO226 Reset type: SYSRSn
1	GPIO225	R/W	0h	Read: Input value for GPIO225 Reset type: SYSRSn
0	GPIO224	R/W	0h	Read: Input value for GPIO224 Reset type: SYSRSn

### 8.7.4 Register to Driverlib Function Mapping

**Table 8-65. GPIO Registers to Driverlib Functions**

File	Driverlib Function
<b>GPACTRL</b>	
gpio.c	GPIO_setQualificationPeriod
<b>GPAQSEL1</b>	
gpio.c	GPIO_setQualificationMode
gpio.c	GPIO_getQualificationMode
<b>GPAQSEL2</b>	
-	See GPAQSEL1
<b>GPAMUX1</b>	
gpio.c	GPIO_setPinConfig
<b>GPAMUX2</b>	
-	See GPAMUX1
<b>GPADIR</b>	
gpio.c	GPIO_setDirectionMode
gpio.c	GPIO_getDirectionMode
<b>GPAPUD</b>	
gpio.c	GPIO_setPadConfig
gpio.c	GPIO_getPadConfig
<b>GPAINV</b>	
gpio.c	GPIO_setPadConfig
gpio.c	GPIO_getPadConfig
<b>GPAODR</b>	
gpio.c	GPIO_setPadConfig
gpio.c	GPIO_getPadConfig
<b>GPAGMUX1</b>	
gpio.c	GPIO_setPinConfig
<b>GPAGMUX2</b>	
-	See GPAGMUX1
<b>GPACSEL1</b>	
gpio.c	GPIO_setMasterCore
<b>GPACSEL2</b>	
-	See GPACSEL1
<b>GPACSEL3</b>	
-	See GPACSEL1
<b>GPACSEL4</b>	
-	See GPACSEL1
<b>GPALOCK</b>	
gpio.h	GPIO_lockPortConfig
gpio.h	GPIO_unlockPortConfig
<b>GPACR</b>	
gpio.h	GPIO_commitPortConfig
<b>GPBCTRL</b>	
-	See GPACTRL
<b>GPBQSEL1</b>	
-	See GPAQSEL1
<b>GPBQSEL2</b>	
-	See GPAQSEL1

**Table 8-65. GPIO Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>GPBMUX1</b>	
-	See GPAMUX1
<b>GPBMUX2</b>	
-	See GPAMUX1
<b>GPBDIR</b>	
-	See GPADIR
<b>GPBPUD</b>	
-	See GPAPUD
<b>GPBINV</b>	
-	See GPAINV
<b>GPBODR</b>	
-	See GPAODR
<b>GPBGMUX1</b>	
-	See GPAGMUX1
<b>GPBGMUX2</b>	
-	See GPAGMUX1
<b>GPBCSEL1</b>	
-	See GPACSEL1
<b>GPBCSEL2</b>	
-	See GPACSEL1
<b>GPBCSEL3</b>	
-	See GPACSEL1
<b>GPBCSEL4</b>	
-	See GPACSEL1
<b>GPBLOCK</b>	
-	See GPALOCK
<b>GPBCR</b>	
-	See GPACR
<b>GPHCTRL</b>	
-	See GPACTRL
<b>GPHQSEL1</b>	
-	See GPAQSEL1
<b>GPHQSEL2</b>	
-	See GPAQSEL1
<b>GPHINV</b>	
-	See GPAINV
<b>GPHLOCK</b>	
-	See GPALOCK
<b>GPHCR</b>	
-	See GPACR
<b>GPADAT</b>	
gpio.h	GPIO_readPin
gpio.h	GPIO_readPortData
gpio.h	GPIO_writePortData
<b>GPASET</b>	
gpio.h	GPIO_writePin
gpio.h	GPIO_setPortPins

**Table 8-65. GPIO Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>GPACLEAR</b>	
gpio.h	GPIO_writePin
gpio.h	GPIO_clearPortPins
<b>GPATOGGLE</b>	
gpio.h	GPIO_togglePin
gpio.h	GPIO_togglePortPins
<b>GPBDAT</b>	
-	See GPADAT
<b>GPBSET</b>	
-	See GPASET
<b>GPBCLEAR</b>	
-	See GPACLEAR
<b>GPBTOGGLE</b>	
-	See GPATOGGLE
<b>GPHDAT</b>	
-	See GPADAT

## Crossbar (X-BAR)

---



---

The crossbars (referred to as X-BAR throughout this document) provide flexibility to connect device inputs, outputs, and internal resources in a variety of configurations. The device contains a total of three X-BARs: the Input X-BAR, the Output X-BAR, and the ePWM X-BAR. Each of the X-BARs is named according to where they take signals. For example, the Input X-BAR brings external signals “in” to the device. The Output X-BAR takes internal signals “out” of the device to a GPIO. The ePWM X-BAR takes signals and brings them to the ePWM modules. You can read more about each of these X-BARs in the following sections.

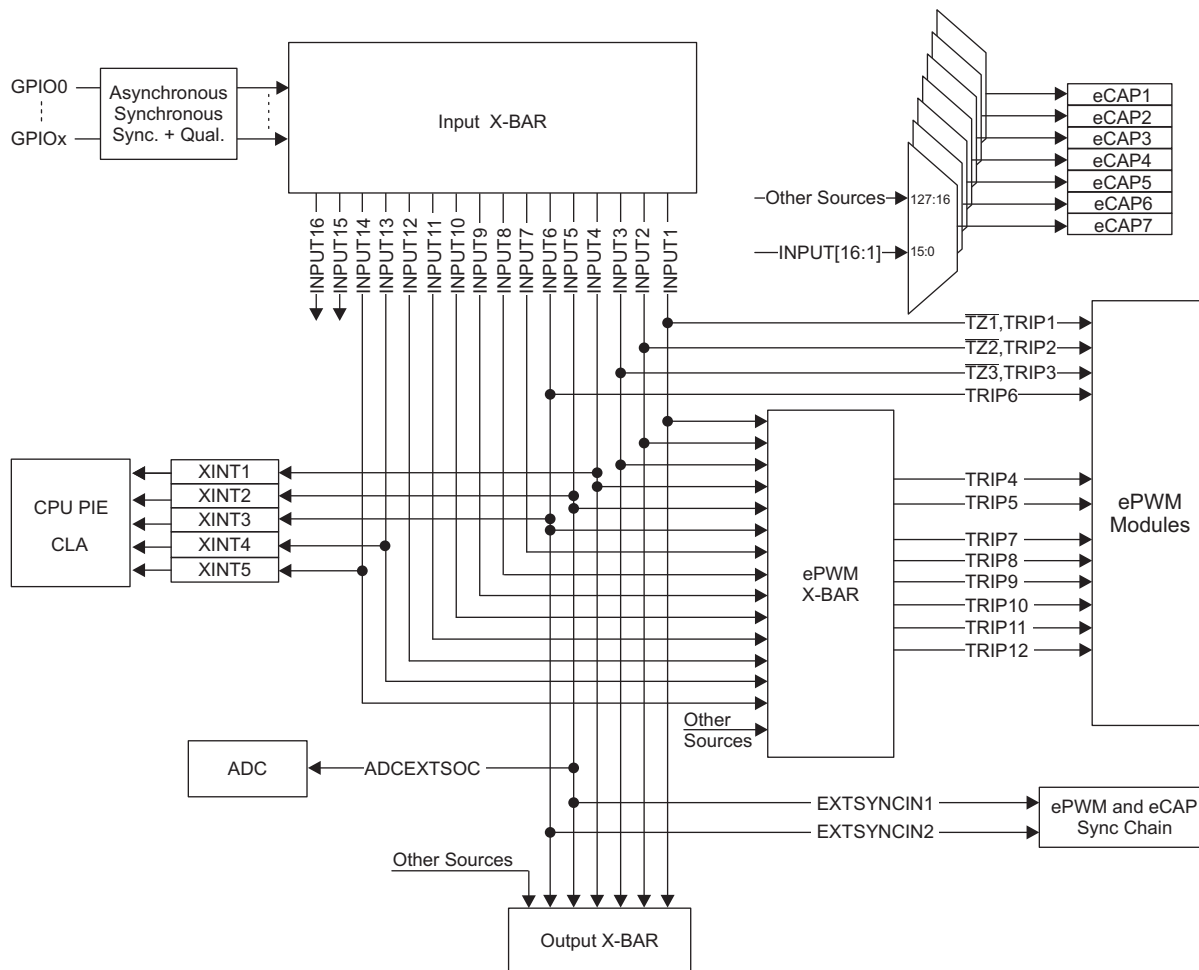
Topic	Page
<b>9.1 Introduction .....</b>	<b>990</b>
<b>9.2 ePWM, CLB, and GPIO Output X-BAR .....</b>	<b>991</b>
<b>9.3 XBAR Registers.....</b>	<b>999</b>

## 9.1 Introduction

On this device, the Input X-BAR is used to route signals from a GPIO to many different IP blocks such as the ADC(s), eCAP(s), ePWM(s), and external interrupts. The Input X-BAR has access to every GPIO and can route each signal to any (or multiple) of the IP blocks previously mentioned. The digital input of AIOs are also available on the Input X-BAR. This flexibility relieves some of the constraints on peripheral muxing by just requiring any GPIO pin to be available. It is important to note that the function selected on the GPIO mux does not affect the Input X-BAR. The Input X-BAR simply connects the signal on the input buffer to the selected destination. Therefore, you can do things such as route the output of one peripheral to another (that is, measure the output of an ePWM with an eCAP for a frequency test).

The Input X-BAR is configured via the INPUTxSELECT registers. The available IP destination(s) for each INPUTx is shown in [Figure 9-1](#). For more information on configuration, see the INPUT\_XBAR\_REGS register definitions at the end of this chapter.

Figure 9-1. Input X-BAR



**Table 9-1. Input X-BAR Destinations**

INPUT	DESTINATIONS
INPUT1	eCAPx, ePWM X-BAR, ePWM[TZ1,TRIP1], Output X-BAR
INPUT2	eCAPx, ePWM X-BAR, ePWM[TZ2,TRIP2], Output X-BAR
INPUT3	eCAPx, ePWM X-BAR, ePWM[TZ3,TRIP3], Output X-BAR
INPUT4	eCAPx, ePWM X-BAR, XINT1, Output X-BAR
INPUT5	eCAPx, ePWM X-BAR, XINT2, ADCEXTSOC, EXTSYNCIN1, Output X-BAR
INPUT6	eCAPx, ePWM X-BAR, XINT3, ePWM[TRIP6], EXTSYNCIN2, Output X-BAR
INPUT7	eCAPx, ePWM X-BAR
INPUT8	eCAPx, ePWM X-BAR
INPUT9	eCAPx, ePWM X-BAR
INPUT10	eCAPx, ePWM X-BAR
INPUT11	eCAPx, ePWM X-BAR
INPUT12	eCAPx, ePWM X-BAR
INPUT13	eCAPx, ePWM X-BAR, XINT4
INPUT14	eCAPx, ePWM X-BAR, XINT5
INPUT15	eCAPx
INPUT16	eCAPx

## 9.2 ePWM, CLB, and GPIO Output X-BAR

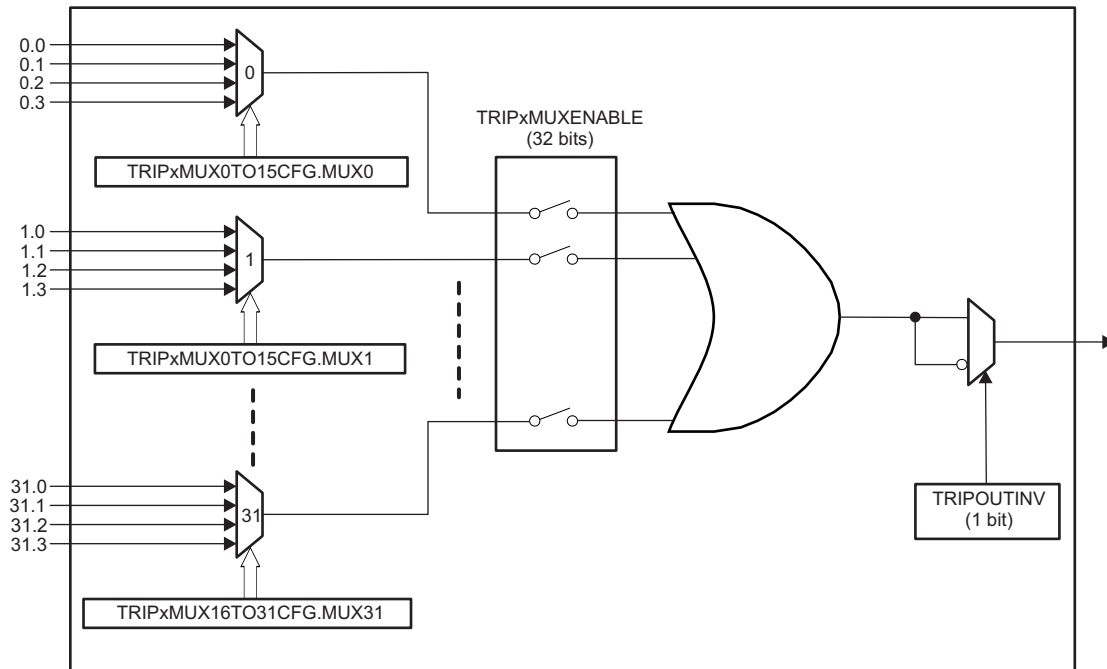
This section describes the ePWM , CLB, and GPIO Output X-BAR

### 9.2.1 ePWM X-BAR

The ePWM X-BAR brings signals to the ePWM modules. Specifically, the ePWM X-BAR is connected to the Digital Compare (DC) submodule of each ePWM module for actions such as tripzones and syncing. Please refer to the *ePWM* chapter for more information on additional ways the DC submodule can be used. [Figure 9-5](#) shows the architecture of the ePWM X-BAR. It is worth noting that the architecture of the ePWM X-BAR is identical to the architecture of the Output X-BAR (with the exception of the output latch).

#### 9.2.1.1 ePWM X-BAR Architecture

The ePWM X-BAR has eight outputs which are routed to each ePWM module. [Figure 18-58](#) represents the architecture of a single output but it is identical to the architecture of all of the other outputs.

**Figure 9-2. ePWM X-BAR Architecture - Single Output**


First, determine the signal(s) which should be passed to the ePWM by referencing [Table 9-2](#). You may select up to one signal per mux (32 total muxes) for each TRIPx output. Select the inputs to each mux via the TRIPxMUX0TO15CFG and TRIPxMUX16TO31CFG registers. In order to pass any signal through to the ePWM, you must also enable the mux in the TRIPxMUXENABLE register. All muxes which are enabled will be logically OR'd before being passed on to the respective TRIPx signal on the ePWM. You may also optionally invert the signal via the TRIPOUTINV register.



**Table 9-2. ePWM X-BAR Mux Configuration Table**

Mux	0	1	2	3
0	CMPSS1.CTRIPH	CMPSS1.CTRIPH_OR_CTRIPL	ADCAEVT1	ECAP1OUT
1	CMPSS1.CTRIPL	INPUTXBAR1	CLB1_OUT4	ADCCEVT1
2	CMPSS2.CTRIPH	CMPSS2.CTRIPH_OR_CTRIPL	ADCAEVT2	ECAP2OUT
3	CMPSS2.CTRIPL	INPUTXBAR2	CLB1_OUT5	ADCCEVT2
4	CMPSS3.CTRIPH	CMPSS3.CTRIPH_OR_CTRIPL	ADCAEVT3	ECAP3OUT
5	CMPSS3.CTRIPL	INPUTXBAR3	CLB2_OUT4	ADCCEVT3
6	CMPSS4.CTRIPH	CMPSS4.CTRIPH_OR_CTRIPL	ADCAEVT4	ECAP4OUT
7	CMPSS4.CTRIPL	INPUTXBAR4	CLB2_OUT5	ADCCEVT4
8	CMPSS5.CTRIPH	CMPSS5.CTRIPH_OR_CTRIPL	ADCBEVT1	ECAP5OUT
9	CMPSS5.CTRIPL	INPUTXBAR5	CLB3_OUT4	Reserved
10	CMPSS6.CTRIPH	CMPSS6.CTRIPH_OR_CTRIPL	ADCBEVT2	ECAP6OUT
11	CMPSS6.CTRIPL	INPUTXBAR6	CLB3_OUT5	Reserved
12	CMPSS7.CTRIPH	CMPSS7.CTRIPH_OR_CTRIPL	ADCBEVT3	ECAP7OUT
13	CMPSS7.CTRIPL	ADCSOCAO	CLB4_OUT4	Reserved
14	Reserved	Reserved	ADCBEVT4	EXTSYNCOUT
15	Reserved	ADCSOCBO	CLB4_OUT5	Reserved
16	SD1FLT1.COMPH	SD1FLT1.COMPH_OR_COMPL	Reserved	Reserved
17	SD1FLT1.COMPL	INPUT7	Reserved	CLAHALT
18	SD1FLT2.COMPH	SD1FLT2.COMPH_OR_COMPL	Reserved	Reserved
19	SD1FLT2.COMPL	INPUT8	Reserved	Reserved
20	SD1FLT3.COMPH	SD1FLT3.COMPH_OR_COMPL	Reserved	Reserved
21	SD1FLT3.COMPL	INPUT9	Reserved	Reserved
22	SD1FLT4.COMPH	SD1FLT4.COMPH_OR_COMPL	Reserved	Reserved
23	SD1FLT4.COMPL	INPUT10	Reserved	Reserved
24	Reserved	Reserved	Reserved	Reserved
25	Reserved	INPUT11	Reserved	Reserved
26	Reserved	Reserved	Reserved	Reserved
27	Reserved	INPUT12	Reserved	Reserved
28	Reserved	Reserved	Reserved	Reserved
29	Reserved	INPUT13	Reserved	Reserved
30	Reserved	Reserved	Reserved	Reserved
31	Reserved	INPUT14	Reserved	Reserved

---

**NOTE:** Please do not use "Reserved" signals in your application.

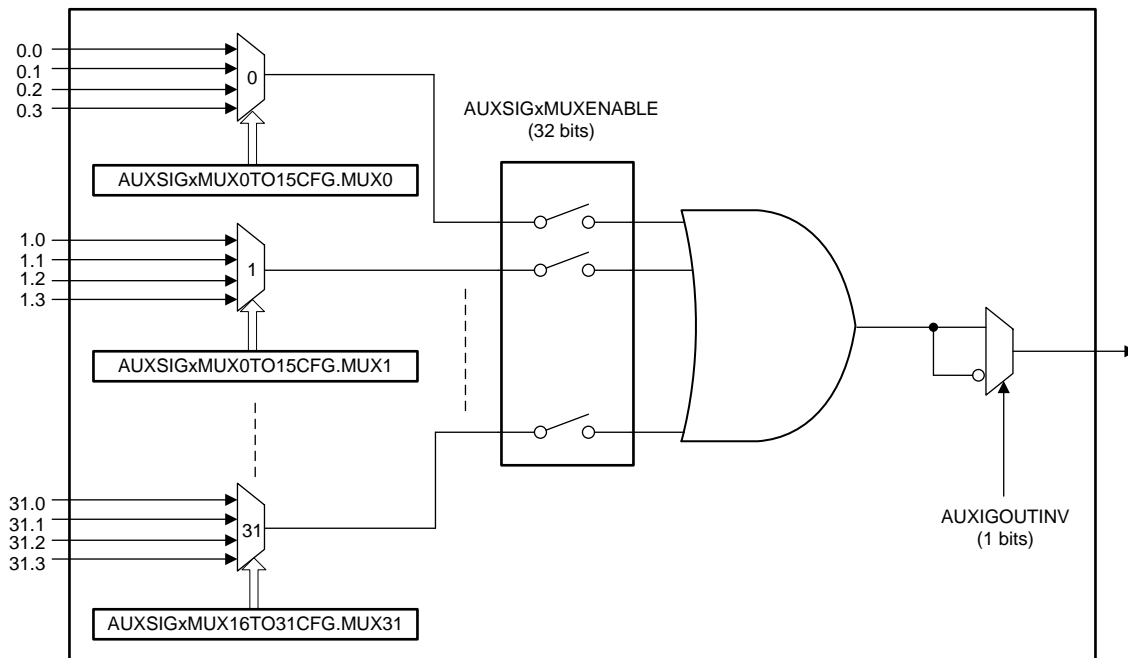
---

## 9.2.2 CLB X-BAR

The CLB X-BAR brings signals to the CLB modules. [Figure 9-5](#) shows the architecture of the CLB X-BAR. It is worth noting that the architecture of the CLB X-BAR is identical to the architecture of the Output X-BAR (with the exception of the output latch).

### 9.2.2.1 CLB X-BAR Architecture

The CLB X-BAR has eight outputs which are routed to each CLB module. [Figure 9-3](#) represents the architecture of a single output but it is identical to the architecture of all of the other outputs.

**Figure 9-3. CLB X-BAR Architecture - Single Output**


First, determine the signal(s) which should be passed to the CLB by referencing [Table 9-3](#). You may select up to one signal per mux (31 total muxes) for each AUXSIGx output. Select the inputs to each mux via the AUXSIGxMUX0TO15CFG and AUXSIGxMUX16TO31CFG registers. In order to pass any signal through to the CLB, you must also enable the mux in the AUXSIGxMUXENABLE register. All muxes which are enabled will be logically OR'd before being passed on to the respective AUXSIGx signal on the CLB. You may also optionally invert the signal via the AUXSIGOUTINV register.

**Table 9-3. CLB X-BAR Mux Configuration Table**

Mux	0	1	2	3
0	CMPSS1.CTRIPOUTH	CMPSS1.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT1	ECAP1OUT
1	CMPSS1.CTRIPOUTL	INPUTXBAR1	CLB1_OUT4	ADCCEVT1
2	CMPSS2.CTRIPOUTH	CMPSS2.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT2	ECAP2OUT
3	CMPSS2.CTRIPOUTL	INPUTXBAR2	CLB1_OUT5	ADCCEVT2
4	CMPSS3.CTRIPOUTH	CMPSS3.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT3	ECAP3OUT
5	CMPSS3.CTRIPOUTL	INPUTXBAR3	CLB2_OUT4	ADCCEVT3
6	CMPSS4.CTRIPOUTH	CMPSS4.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT4	ECAP4OUT
7	CMPSS4.CTRIPOUTL	INPUTXBAR4	CLB2_OUT5	ADCCEVT4
8	CMPSS5.CTRIPOUTH	CMPSS5.CTRIPOUTH_OR_CTRIPOUTL	ADCBEVT1	ECAP5OUT
9	CMPSS5.CTRIPOUTL	INPUTXBAR5	CLB3_OUT4	Reserved
10	CMPSS6.CTRIPOUTH	CMPSS6.CTRIPOUTH_OR_CTRIPOUTL	ADCBEVT2	ECAP6OUT
11	CMPSS6.CTRIPOUTL	INPUTXBAR6	CLB3_OUT5	Reserved
12	CMPSS7.CTRIPOUTH	CMPSS7.CTRIPOUTH_OR_CTRIPOUTL	ADCBEVT3	ECAP7OUT
13	CMPSS7.CTRIPOUTL	ADCSOCAO	CLB4_OUT4	Reserved
14	Reserved	Reserved	ADCBEVT4	EXTSYNCOUT
15	Reserved	ADCSOCBO	CLB4_OUT5	Reserved
16	SD1FLT1.COMPH	SD1FLT1.COMPH_OR_COMPL	Reserved	Reserved
17	SD1FLT1.COMPL	Reserved	Reserved	CLAHALT
18	SD1FLT2.COMPH	SD1FLT2.COMPH_OR_COMPL	Reserved	Reserved
19	SD1FLT2.COMPL	Reserved	Reserved	Reserved

**Table 9-3. CLB X-BAR Mux Configuration Table (continued)**

<b>Mux</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>
20	SD1FLT3.COMPH	SD1FLT3.COMPH_OR_COMPL	Reserved	Reserved
21	SD1FLT3.COMPL	Reserved	Reserved	Reserved
22	SD1FLT4.COMPH	SD1FLT4.COMPH_OR_COMPL	Reserved	Reserved
23	SD1FLT4.COMPL	Reserved	Reserved	Reserved

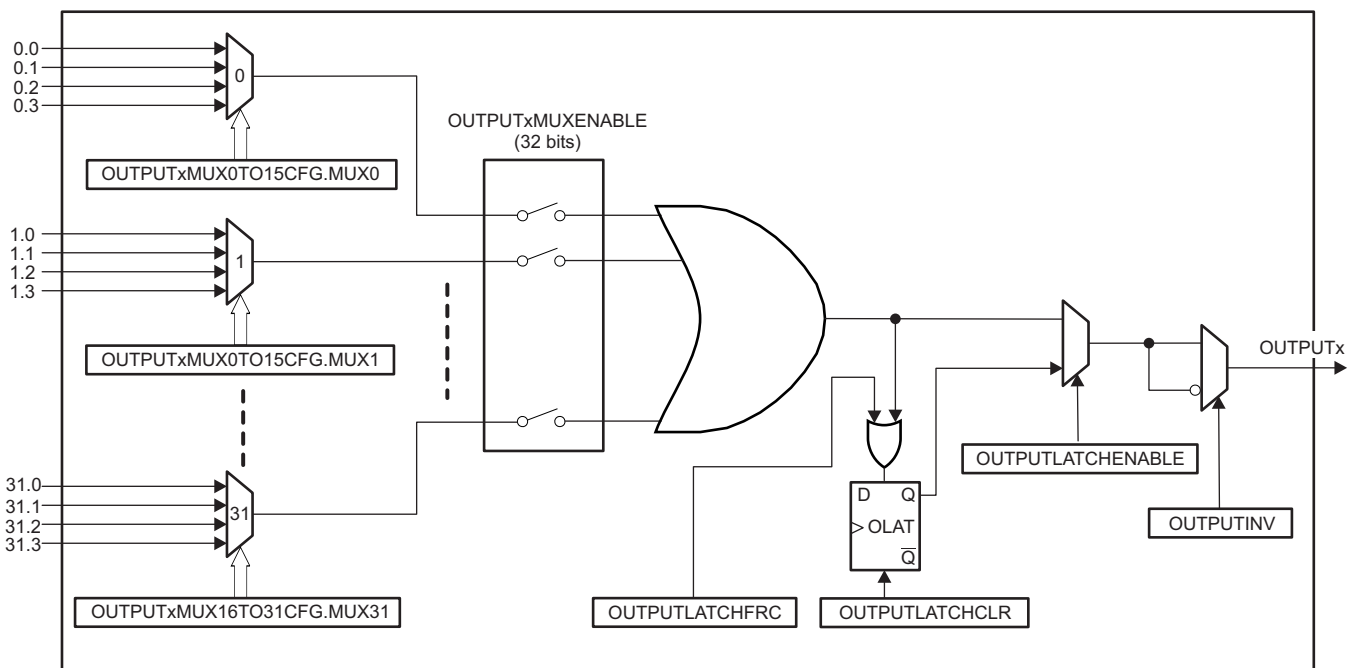
## 9.2.3 GPIO Output X-BAR

The GPIO Output X-BAR takes signals from inside the device and brings them out to a GPIO. [Figure 9-5](#) shows the architecture of the GPIO Output X-BAR. The signals which are available to bring to the GPIO are listed in [Table 9-4](#). The X-BAR contains eight outputs and each will contain at least one position on the GPIO mux, denoted as OUTPUTXBARx. The X-BAR allows the selection of a single signal or a logical OR of up to 32 signals.

### 9.2.3.1 GPIO Output X-BAR Architecture

The Output X-BAR has eight outputs which are routed to the GPIO module. [Figure 9-4](#) represents the architecture of a single output, but it is identical to the architecture of all of the other outputs. It is worth noting that the architecture of the Output X-BAR (with the exception of the output latch) is identical to the architecture of the ePWM X-BAR.

**Figure 9-4. GPIO Output X-BAR Architecture**



First, determine the signal(s) which should be passed to the GPIO by referencing [Table 9-4](#). You may select up to one signal per mux (32 total muxes) for each OUTPUTXBARx output. Select the inputs to each mux via the OUTPUTxMUX0TO15CFG and OUTPUTxMUX16TO31CFG registers.

In order to pass any signal through to the GPIO, you must also enable the mux in the OUTPUTxMUXENABLE register. All muxes which are enabled will be logically OR'd before being passed on to the respective OUTPUTx signal on the GPIO module. You may also optionally invert the signal via the OUTPUTINV register. The signal will only be seen on the GPIO if the proper OUTPUTx muxing options are selected via the GpioCtrlRegs.GPxMUX and GpioCtrlRegs.GPxGMUX registers.

**Table 9-4. Output X-BAR Mux Configuration Table**

Mux	0	1	2	3
0	CMPSS1.CTRIPOUTH	CMPSS1.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT1	ECAP1OUT
1	CMPSS1.CTRIPOUTL	INPUTXBAR1	CLB1_OUT4	ADCCEVT1
2	CMPSS2.CTRIPOUTH	CMPSS2.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT2	ECAP2OUT
3	CMPSS2.CTRIPOUTL	INPUTXBAR2	CLB1_OUT5	ADCCEVT2
4	CMPSS3.CTRIPOUTH	CMPSS3.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT3	ECAP3OUT
5	CMPSS3.CTRIPOUTL	INPUTXBAR3	CLB2_OUT4	ADCCEVT3
6	CMPSS4.CTRIPOUTH	CMPSS4.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT4	ECAP4OUT
7	CMPSS4.CTRIPOUTL	INPUTXBAR4	CLB2_OUT5	ADCCEVT4
8	CMPSS5.CTRIPOUTH	CMPSS5.CTRIPOUTH_OR_CTRIPOUTL	ADCBEVT1	ECAP5OUT
9	CMPSS5.CTRIPOUTL	INPUTXBAR5	CLB3_OUT4	Reserved
10	CMPSS6.CTRIPOUTH	CMPSS6.CTRIPOUTH_OR_CTRIPOUTL	ADCBEVT2	ECAP6OUT
11	CMPSS6.CTRIPOUTL	INPUTXBAR6	CLB3_OUT5	Reserved
12	CMPSS7.CTRIPOUTH	CMPSS7.CTRIPOUTH_OR_CTRIPOUTL	ADCBEVT3	ECAP7OUT
13	CMPSS7.CTRIPOUTL	ADCSOCAO	CLB4_OUT4	Reserved
14	Reserved	Reserved	ADCBEVT4	EXTSYNCOUT
15	Reserved	ADCSOCBO	CLB4_OUT5	Reserved
16	SD1FLT1.COMPH	SD1FLT1.COMPH_OR_COMPL	Reserved	Reserved
17	SD1FLT1.COMPL	Reserved	Reserved	CLAHALT
18	SD1FLT2.COMPH	SD1FLT2.COMPH_OR_COMPL	Reserved	Reserved
19	SD1FLT2.COMPL	Reserved	Reserved	Reserved
20	SD1FLT3.COMPH	SD1FLT3.COMPH_OR_COMPL	Reserved	Reserved
21	SD1FLT3.COMPL	Reserved	Reserved	Reserved
22	SD1FLT4.COMPH	SD1FLT4.COMPH_OR_COMPL	Reserved	Reserved
23	SD1FLT4.COMPL	Reserved	Reserved	Reserved

---

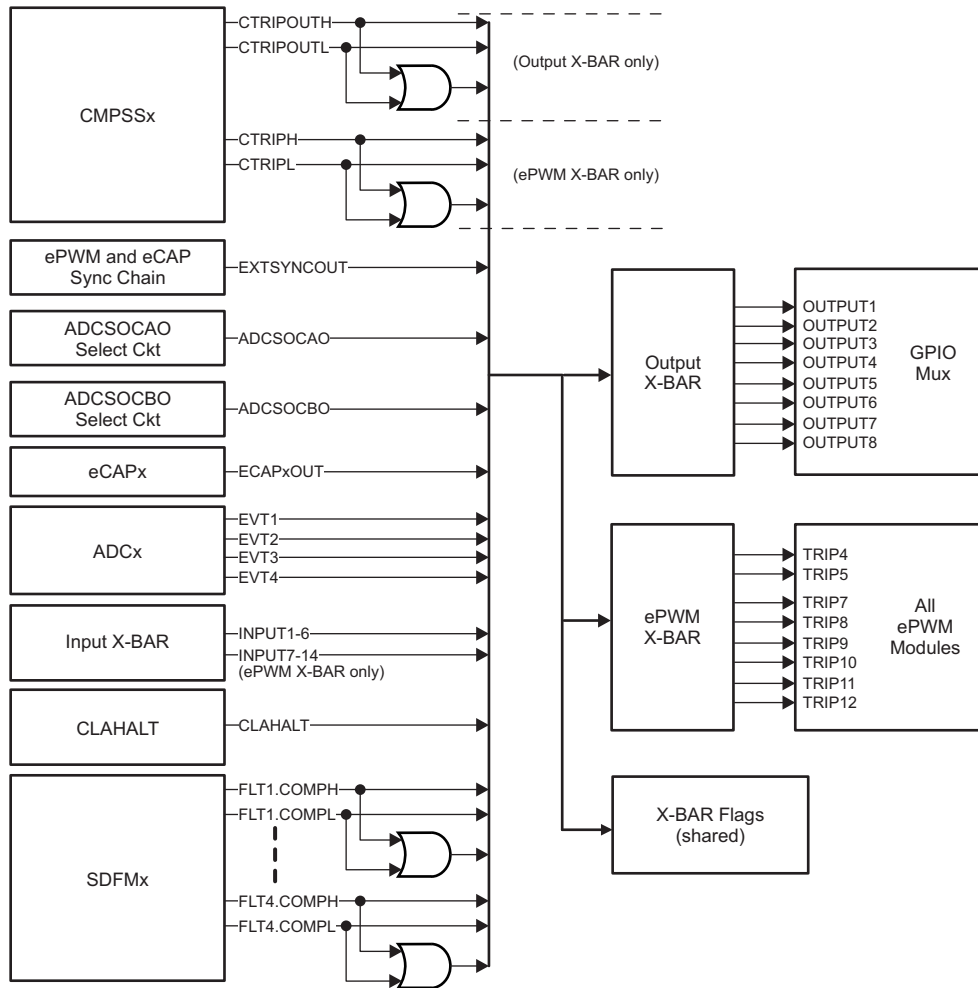
**NOTE:** Please do not use "Reserved" signals in your application.

---

### 9.2.4 X-BAR Flags

With the exception of the CMPSS signals, the ePWM X-BAR and the Output X-BAR have all of the same input signals. Due to the inputs being similar, the ePWM X-BAR and Output X-BAR leverage a single set of input flags to indicate which input signals have been triggered. This allows software to check the input flags when an event occurs. See [Figure 9-5](#) for more information. There is a bit allocated for each input signal in one of the XBARFLGx registers. The flag will remain set until cleared through the appropriate XBARCLR register.

Figure 9-5. ePWM and Output X-BARs Sources



## 9.3 XBAR Registers

This section describes the Crossbar registers.

### 9.3.1 XBAR Base Addresses

**Table 9-5. System Control Base Address Table**

Device Registers	Register Name	Start Address	End Address
InputXbarRegs	INPUT_XBAR_REGS	0x0000_7900	0x0000_791F
XbarRegs	XBAR_REGS	0x0000_7920	0x0000_793F
EpwmXbarRegs	EPWM_XBAR_REGS	0x0000_7A00	0x0000_7A3F
OutputXbarRegs	OUTPUT_XBAR_REGS	0x0000_7A80	0x0000_7ABF
ClbXbarRegs	CLB_XBAR_REGS	0x0000_7A40	0x0000_7A7F

### 9.3.2 XBAR\_REGS Registers

Table 9-6 lists the XBAR\_REGS registers. All register offset addresses not listed in Table 9-6 should be considered as reserved locations and the register contents should not be modified.

**Table 9-6. XBAR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	XBARFLG1	X-Bar Input Flag Register 1		<a href="#">Go</a>
2h	XBARFLG2	X-Bar Input Flag Register 2		<a href="#">Go</a>
4h	XBARFLG3	X-Bar Input Flag Register 3		<a href="#">Go</a>
6h	XBARFLG4	X-Bar Input Flag Register 4		<a href="#">Go</a>
8h	XBARCLR1	X-Bar Input Flag Clear Register 1		<a href="#">Go</a>
Ah	XBARCLR2	X-Bar Input Flag Clear Register 2		<a href="#">Go</a>
Ch	XBARCLR3	X-Bar Input Flag Clear Register 3		<a href="#">Go</a>
Eh	XBARCLR4	X-Bar Input Flag Clear Register 4		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 9-7 shows the codes that are used for access types in this section.

**Table 9-7. XBAR\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.



### 9.3.2.1 XBARFLG1 Register (Offset = 0h) [reset = 0h]

XBARFLG1 is shown in [Figure 9-6](#) and described in [Table 9-8](#).

Return to the [Summary Table](#).

This register is used to flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.

- 1: Corresponding Input was triggered
- 0: Corresponding Input was not triggered

**Figure 9-6. XBARFLG1 Register**

31	30	29	28	27	26	25	24
CMPSS8_CTRL_POUTH	CMPSS8_CTRL_POU TL	CMPSS7_CTRL_POUTH	CMPSS7_CTRL_POU TL	CMPSS6_CTRL_POUTH	CMPSS6_CTRL_POU TL	CMPSS5_CTRL_POUTH	CMPSS5_CTRL_POU TL
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
CMPSS4_CTRL_POUTH	CMPSS4_CTRL_POU TL	CMPSS3_CTRL_POUTH	CMPSS3_CTRL_POU TL	CMPSS2_CTRL_POUTH	CMPSS2_CTRL_POU TL	CMPSS1_CTRL_POUTH	CMPSS1_CTRL_POU TL
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
CMPSS8_CTRL_PH	CMPSS8_CTRL_PL	CMPSS7_CTRL_PH	CMPSS7_CTRL_PL	CMPSS6_CTRL_PH	CMPSS6_CTRL_PL	CMPSS5_CTRL_PH	CMPSS5_CTRL_PL
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
CMPSS4_CTRL_PH	CMPSS4_CTRL_PL	CMPSS3_CTRL_PH	CMPSS3_CTRL_PL	CMPSS2_CTRL_PH	CMPSS2_CTRL_PL	CMPSS1_CTRL_PH	CMPSS1_CTRL_PL
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 9-8. XBARFLG1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	CMPSS8_CTRLPOUTH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS8_CTRLPOUTH input was triggered 0: CMPSS8_CTRLPOUTH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
30	CMPSS8_CTRLPOUTL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS8_CTRLPOUTL input was triggered 0: CMPSS8_CTRLPOUTL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
29	CMPSS7_CTRLPOUTH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS7_CTRLPOUTH input was triggered 0: CMPSS7_CTRLPOUTH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-8. XBARFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28	CMPSS7_CTRIPOUTL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS7_CTRIPOUTL input was triggered 0: CMPSS7_CTRIPOUTL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
27	CMPSS6_CTRIPOUTH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS6_CTRIPOUTH input was triggered 0: CMPSS6_CTRIPOUTH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
26	CMPSS6_CTRIPOUTL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS6_CTRIPOUTL input was triggered 0: CMPSS6_CTRIPOUTL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
25	CMPSS5_CTRIPOUTH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS5_CTRIPOUTH input was triggered 0: CMPSS5_CTRIPOUTH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
24	CMPSS5_CTRIPOUTL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS5_CTRIPOUTL input was triggered 0: CMPSS5_CTRIPOUTL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
23	CMPSS4_CTRIPOUTH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS4_CTRIPOUTH input was triggered 0: CMPSS4_CTRIPOUTH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-8. XBARFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	CMPSS4_CTRIPOUTL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS4_CTRIPOUTL input was triggered 0: CMPSS4_CTRIPOUTL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
21	CMPSS3_CTRIPOUTH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS3_CTRIPOUTH input was triggered 0: CMPSS3_CTRIPOUTH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
20	CMPSS3_CTRIPOUTL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS3_CTRIPOUTL input was triggered 0: CMPSS3_CTRIPOUTL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
19	CMPSS2_CTRIPOUTH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS2_CTRIPOUTH input was triggered 0: CMPSS2_CTRIPOUTH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
18	CMPSS2_CTRIPOUTL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS2_CTRIPOUTL input was triggered 0: CMPSS2_CTRIPOUTL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
17	CMPSS1_CTRIPOUTH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS1_CTRIPOUTH input was triggered 0: CMPSS1_CTRIPOUTH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-8. XBARFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	CMPSS1_CTRIPOUTL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS1_CTRIPOUTL input was triggered 0: CMPSS1_CTRIPOUTL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
15	CMPSS8_CTRIPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS8_CTRIPH input was triggered 0: CMPSS8_CTRIPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
14	CMPSS8_CTRIPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS8_CTRIPL input was triggered 0: CMPSS8_CTRIPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
13	CMPSS7_CTRIPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS7_CTRIPH input was triggered 0: CMPSS7_CTRIPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
12	CMPSS7_CTRIPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS7_CTRIPL input was triggered 0: CMPSS7_CTRIPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
11	CMPSS6_CTRIPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS6_CTRIPH input was triggered 0: CMPSS6_CTRIPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-8. XBARFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	CMPSS6_CTRIPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS6_CTRIPL input was triggered 0: CMPSS6_CTRIPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
9	CMPSS5_CTRIPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS5_CTRIPH input was triggered 0: CMPSS5_CTRIPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
8	CMPSS5_CTRIPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS5_CTRIPL input was triggered 0: CMPSS5_CTRIPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
7	CMPSS4_CTRIPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS4_CTRIPH input was triggered 0: CMPSS4_CTRIPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
6	CMPSS4_CTRIPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS4_CTRIPL input was triggered 0: CMPSS4_CTRIPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
5	CMPSS3_CTRIPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS3_CTRIPH input was triggered 0: CMPSS3_CTRIPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-8. XBARFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	CMPSS3_CTRIPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS3_CTRIPL input was triggered 0: CMPSS3_CTRIPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
3	CMPSS2_CTRIPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS2_CTRIPH input was triggered 0: CMPSS2_CTRIPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
2	CMPSS2_CTRIPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS2_CTRIPL input was triggered 0: CMPSS2_CTRIPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
1	CMPSS1_CTRIPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS1_CTRIPH input was triggered 0: CMPSS1_CTRIPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
0	CMPSS1_CTRIPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CMPSS1_CTRIPL input was triggered 0: CMPSS1_CTRIPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

### 9.3.2.2 XBARFLG2 Register (Offset = 2h) [reset = 0h]

XBARFLG2 is shown in [Figure 9-7](#) and described in [Table 9-9](#).

Return to the [Summary Table](#).

This register is used to flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.

- 1: Corresponding Input was triggered
- 0: Corresponding Input was not triggered

**Figure 9-7. XBARFLG2 Register**

31		30		29		28		27		26		25		24	
ADCCEVT1	ADCBEVT4	ADCBEVT3	ADCBEVT2	ADCBEVT1	ADCAEVT4	ADCAEVT3	ADCAEVT2	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23		22		21		20		19		18		17		16	
ADCAEVT1	EXTSYNCOUT	ECAP6_OUT	ECAP5_OUT	ECAP4_OUT	ECAP3_OUT	ECAP2_OUT	ECAP1_OUT	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15		14		13		12		11		10		9		8	
INPUT14	INPUT13	INPUT12	INPUT11	INPUT10	INPUT9	INPUT8	INPUT7	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7		6		5		4		3		2		1		0	
ADCSOCB	ADCSOCA	INPUT6	INPUT5	INPUT4	INPUT3	INPUT2	INPUT1	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 9-9. XBARFLG2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	ADCCEVT1	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCCEVT1 input was triggered 0: ADCCEVT1 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
30	ADCBEVT4	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCBEVT4 input was triggered 0: ADCBEVT4 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
29	ADCBEVT3	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCBEVT3 input was triggered 0: ADCBEVT3 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-9. XBARFLG2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28	ADCBEVT2	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCBEVT2 input was triggered 0: ADCBEVT2 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
27	ADCBEVT1	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCBEVT1 input was triggered 0: ADCBEVT1 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
26	ADCAEVT4	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCAEVT4 input was triggered 0: ADCAEVT4 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
25	ADCAEVT3	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCAEVT3 input was triggered 0: ADCAEVT3 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
24	ADCAEVT2	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCAEVT2 input was triggered 0: ADCAEVT2 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
23	ADCAEVT1	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCAEVT1 input was triggered 0: ADCAEVT1 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>



**Table 9-9. XBARFLG2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	EXTSYNCOUT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: EXTSYNCOUT input was triggered 0: EXTSYNCOUT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
21	ECAP6_OUT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ECAP6_OUT input was triggered 0: ECAP6_OUT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
20	ECAP5_OUT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ECAP5_OUT input was triggered 0: ECAP5_OUT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
19	ECAP4_OUT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ECAP4_OUT input was triggered 0: ECAP4_OUT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
18	ECAP3_OUT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ECAP3_OUT input was triggered 0: ECAP3_OUT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
17	ECAP2_OUT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ECAP2_OUT input was triggered 0: ECAP2_OUT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-9. XBARFLG2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	ECAP1_OUT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ECAP1_OUT input was triggered 0: ECAP1_OUT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
15	INPUT14	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT14 input was triggered 0: INPUT14 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
14	INPUT13	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT13 input was triggered 0: INPUT13 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
13	INPUT12	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT12 input was triggered 0: INPUT12 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
12	INPUT11	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT11 input was triggered 0: INPUT11 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
11	INPUT10	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT10 input was triggered 0: INPUT10 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-9. XBARFLG2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	INPUT9	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT9 input was triggered 0: INPUT9 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
9	INPUT8	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT8 input was triggered 0: INPUT8 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
8	INPUT7	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT7 input was triggered 0: INPUT7 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
7	ADCSOCB	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCSOCB input was triggered 0: ADCSOCB Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
6	ADCSOCA	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCSOCA input was triggered 0: ADCSOCA Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
5	INPUT6	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT6 input was triggered 0: INPUT6 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-9. XBARFLG2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	INPUT5	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT5 input was triggered 0: INPUT5 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
3	INPUT4	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT4 input was triggered 0: INPUT4 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
2	INPUT3	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT3 input was triggered 0: INPUT3 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
1	INPUT2	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT2 input was triggered 0: INPUT2 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
0	INPUT1	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: INPUT1 input was triggered 0: INPUT1 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

### 9.3.2.3 XBARFLG3 Register (Offset = 4h) [reset = 0h]

XBARFLG3 is shown in [Figure 9-8](#) and described in [Table 9-10](#).

Return to the [Summary Table](#).

This register is used to flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.

- 1: Corresponding Input was triggered
- 0: Corresponding Input was not triggered

**Figure 9-8. XBARFLG3 Register**

31	30	29	28	27	26	25	24
SD1FLT4_DRINT	SD1FLT4_COMPZ	SD1FLT3_DRINT	SD1FLT3_COMPZ	SD1FLT2_DRINT	SD1FLT2_COMPZ	SD1FLT1_DRINT	SD1FLT1_COMPZ
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
ECAP7_OUT	SD2FLT4_COMPH	SD2FLT4_COMPL	SD2FLT3_COMPH	SD2FLT3_COMPL	SD2FLT2_COMPH	SD2FLT2_COMPL	SD2FLT1_COMPH
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
SD2FLT1_COMPL	SD1FLT4_COMPH	SD1FLT4_COMPL	SD1FLT3_COMPH	SD1FLT3_COMPL	SD1FLT2_COMPH	SD1FLT2_COMPL	SD1FLT1_COMPH
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
SD1FLT1_COMPL					ADCCEVT4	ADCCEVT3	ADCCEVT2
R-0h					R-0h	R-0h	R-0h

**Table 9-10. XBARFLG3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	SD1FLT4_DRINT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT4_DRINT input was triggered 0: SD1FLT4_DRINT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
30	SD1FLT4_COMPZ	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT4_COMPZ input was triggered 0: SD1FLT4_COMPZ Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
29	SD1FLT3_DRINT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT3_DRINT input was triggered 0: SD1FLT3_DRINT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-10. XBARFLG3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28	SD1FLT3_COMPZ	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT3_COMPZ input was triggered 0: SD1FLT3_COMPZ Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
27	SD1FLT2_DRINT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT2_DRINT input was triggered 0: SD1FLT2_DRINT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
26	SD1FLT2_COMPZ	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT2_COMPZ input was triggered 0: SD1FLT2_COMPZ Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
25	SD1FLT1_DRINT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT1_DRINT input was triggered 0: SD1FLT1_DRINT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
24	SD1FLT1_COMPZ	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT1_COMPZ input was triggered 0: SD1FLT1_COMPZ Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
23	ECAP7_OUT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ECAP7_OUT input was triggered 0: ECAP7_OUT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-10. XBARFLG3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	SD2FLT4_COMPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT4_COMPH input was triggered 0: SD2FLT4_COMPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
21	SD2FLT4_COMPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT4_COMPL input was triggered 0: SD2FLT4_COMPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
20	SD2FLT3_COMPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT3_COMPH input was triggered 0: SD2FLT3_COMPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
19	SD2FLT3_COMPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT3_COMPL input was triggered 0: SD2FLT3_COMPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
18	SD2FLT2_COMPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT2_COMPH input was triggered 0: SD2FLT2_COMPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
17	SD2FLT2_COMPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT2_COMPL input was triggered 0: SD2FLT2_COMPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-10. XBARFLG3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	SD2FLT1_COMPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT1_COMPH input was triggered 0: SD2FLT1_COMPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
15	SD2FLT1_COMPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT1_COMPL input was triggered 0: SD2FLT1_COMPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
14	SD1FLT4_COMPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT4_COMPH input was triggered 0: SD1FLT4_COMPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
13	SD1FLT4_COMPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT4_COMPL input was triggered 0: SD1FLT4_COMPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
12	SD1FLT3_COMPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT3_COMPH input was triggered 0: SD1FLT3_COMPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
11	SD1FLT3_COMPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT3_COMPL input was triggered 0: SD1FLT3_COMPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>



**Table 9-10. XBARFLG3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	SD1FLT2_COMPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT2_COMPH input was triggered 0: SD1FLT2_COMPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
9	SD1FLT2_COMPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT2_COMPL input was triggered 0: SD1FLT2_COMPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
8	SD1FLT1_COMPH	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT1_COMPH input was triggered 0: SD1FLT1_COMPH Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
7	SD1FLT1_COMPL	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD1FLT1_COMPL input was triggered 0: SD1FLT1_COMPL Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	ADCCEVT4	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCCEVT4 input was triggered 0: ADCCEVT4 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
1	ADCCEVT3	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCCEVT3 input was triggered 0: ADCCEVT3 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-10. XBARFLG3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	ADCCEVT2	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: ADCCEVT2 input was triggered 0: ADCCEVT2 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

### 9.3.2.4 XBARFLG4 Register (Offset = 6h) [reset = 0h]

XBARFLG4 is shown in [Figure 9-9](#) and described in [Table 9-11](#).

Return to the [Summary Table](#).

This register is used to flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.

- 1: Corresponding Input was triggered
- 0: Corresponding Input was not triggered

**Figure 9-9. XBARFLG4 Register**

31		30		29		28		27		26		25		24	
CLAHALT															
R-0h															
23		22		21		20		19		18		17		16	
CLB4_OUT5		CLB4_OUT4		CLB3_OUT5		CLB3_OUT4		CLB2_OUT5		CLB2_OUT4		CLB1_OUT5		CLB1_OUT4	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h	
15		14		13		12		11		10		9		8	
CLB8_5_1		CLB8_4_1		CLB7_5_1		CLB7_4_1		MCANA_FEVT 2		MCANA_FEVT 1		MCANA_FEVT 0		EMAC_PPS0	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h	
7		6		5		4		3		2		1		0	
SD2FLT4_DRI NT		SD2FLT4_CO MPZ		SD2FLT3_DRI NT		SD2FLT3_CO MPZ		SD2FLT2_DRI NT		SD2FLT2_CO MPZ		SD2FLT1_DRI NT		SD2FLT1_CO MPZ	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h	

**Table 9-11. XBARFLG4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	CLAHALT	R	0h	This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.  1: CLAHALT input was triggered 0: CLAHALT Input was not triggered  Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
30	RESERVED	R	0h	Reserved
29	RESERVED	R	0h	Reserved
28	RESERVED	R	0h	Reserved
27	RESERVED	R	0h	Reserved
26	RESERVED	R	0h	Reserved
25	RESERVED	R	0h	Reserved
24	RESERVED	R	0h	Reserved
23	CLB4_OUT5	R	0h	This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.  1: CLB4_OUT5 input was triggered 0: CLB4_OUT5 Input was not triggered  Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn

**Table 9-11. XBARFLG4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	CLB4_OUT4	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB4_OUT4 input was triggered 0: CLB4_OUT4 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
21	CLB3_OUT5	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB3_OUT5 input was triggered 0: CLB3_OUT5 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
20	CLB3_OUT4	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB3_OUT4 input was triggered 0: CLB3_OUT4 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
19	CLB2_OUT5	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB2_OUT5 input was triggered 0: CLB2_OUT5 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
18	CLB2_OUT4	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB2_OUT4 input was triggered 0: CLB2_OUT4 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
17	CLB1_OUT5	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB1_OUT5 input was triggered 0: CLB1_OUT5 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-11. XBARFLG4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	CLB1_OUT4	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB1_OUT4 input was triggered 0: CLB1_OUT4 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
15	CLB8_5_1	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB8_5_1 input was triggered 0: CLB8_5_1 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
14	CLB8_4_1	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB8_4_1 input was triggered 0: CLB8_4_1 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
13	CLB7_5_1	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB7_5_1 input was triggered 0: CLB7_5_1 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
12	CLB7_4_1	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: CLB7_4_1 input was triggered 0: CLB7_4_1 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
11	MCANA_FEVT2	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: MCANA_FEVT2 input was triggered 0: MCANA_FEVT2 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-11. XBARFLG4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	MCANA_FEVT1	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: MCANA_FEVT1 input was triggered 0: MCANA_FEVT1 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
9	MCANA_FEVT0	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: MCANA_FEVT0 input was triggered 0: MCANA_FEVT0 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
8	EMAC_PPS0	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: EMAC_PPS0 input was triggered 0: EMAC_PPS0 Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
7	SD2FLT4_DRINT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT4_DRINT input was triggered 0: SD2FLT4_DRINT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
6	SD2FLT4_COMPZ	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT4_COMPZ input was triggered 0: SD2FLT4_COMPZ Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
5	SD2FLT3_DRINT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT3_DRINT input was triggered 0: SD2FLT3_DRINT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

**Table 9-11. XBARFLG4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	SD2FLT3_COMPZ	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT3_COMPZ input was triggered 0: SD2FLT3_COMPZ Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
3	SD2FLT2_DRINT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT2_DRINT input was triggered 0: SD2FLT2_DRINT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
2	SD2FLT2_COMPZ	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT2_COMPZ input was triggered 0: SD2FLT2_COMPZ Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
1	SD2FLT1_DRINT	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT1_DRINT input was triggered 0: SD2FLT1_DRINT Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
0	SD2FLT1_COMPZ	R	0h	<p>This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.</p> <p>1: SD2FLT1_COMPZ input was triggered 0: SD2FLT1_COMPZ Input was not triggered</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

### 9.3.2.5 XBARCLR1 Register (Offset = 8h) [reset = 0h]

XBARCLR1 is shown in [Figure 9-10](#) and described in [Table 9-12](#).

Return to the [Summary Table](#).

This register is used to clear the flag(s) in the XBARFLG1 register.

1: Clears the corresponding bit in the XBARFLG1 register.

0: Writing 0 has no effect

**Figure 9-10. XBARCLR1 Register**

31	30	29	28	27	26	25	24
CMPSS8_CTRI POUTH	CMPSS8_CTRI POUTL	CMPSS7_CTRI POUTH	CMPSS7_CTRI POUTL	CMPSS6_CTRI POUTH	CMPSS6_CTRI POUTL	CMPSS5_CTRI POUTH	CMPSS5_CTRI POUTL
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
23	22	21	20	19	18	17	16
CMPSS4_CTRI POUTH	CMPSS4_CTRI POUTL	CMPSS3_CTRI POUTH	CMPSS3_CTRI POUTL	CMPSS2_CTRI POUTH	CMPSS2_CTRI POUTL	CMPSS1_CTRI POUTH	CMPSS1_CTRI POUTL
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
15	14	13	12	11	10	9	8
CMPSS8_CTRI PH	CMPSS8_CTRI PL	CMPSS7_CTRI PH	CMPSS7_CTRI PL	CMPSS6_CTRI PH	CMPSS6_CTRI PL	CMPSS5_CTRI PH	CMPSS5_CTRI PL
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
CMPSS4_CTRI PH	CMPSS4_CTRI PL	CMPSS3_CTRI PH	CMPSS3_CTRI PL	CMPSS2_CTRI PH	CMPSS2_CTRI PL	CMPSS1_CTRI PH	CMPSS1_CTRI PL
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 9-12. XBARCLR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	CMPSS8_CTRIPOUTH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS8_CTRIPOUTH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
30	CMPSS8_CTRIPOUTL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS8_CTRIPOUTL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
29	CMPSS7_CTRIPOUTH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS7_CTRIPOUTH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
28	CMPSS7_CTRIPOUTL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS7_CTRIPOUTL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
27	CMPSS6_CTRIPOUTH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS6_CTRIPOUTH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
26	CMPSS6_CTRIPOUTL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS6_CTRIPOUTL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn



**Table 9-12. XBARCLR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25	CMPSS5_CTRIPOUTH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS5_CTRIPOUTH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
24	CMPSS5_CTRIPOUTL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS5_CTRIPOUTL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
23	CMPSS4_CTRIPOUTH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS4_CTRIPOUTH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
22	CMPSS4_CTRIPOUTL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS4_CTRIPOUTL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
21	CMPSS3_CTRIPOUTH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS3_CTRIPOUTH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
20	CMPSS3_CTRIPOUTL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS3_CTRIPOUTL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
19	CMPSS2_CTRIPOUTH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS2_CTRIPOUTH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
18	CMPSS2_CTRIPOUTL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS2_CTRIPOUTL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
17	CMPSS1_CTRIPOUTH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS1_CTRIPOUTH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
16	CMPSS1_CTRIPOUTL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS1_CTRIPOUTL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
15	CMPSS8_CTRIPH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS8_CTRIPH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
14	CMPSS8_CTRIPL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS8_CTRIPL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
13	CMPSS7_CTRIPH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS7_CTRIPH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

**Table 9-12. XBARCLR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	CMPSS7_CTRIPL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS7_CTRIPL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
11	CMPSS6_CTRIPH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS6_CTRIPH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
10	CMPSS6_CTRIPL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS6_CTRIPL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
9	CMPSS5_CTRIPH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS5_CTRIPH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
8	CMPSS5_CTRIPL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS5_CTRIPL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
7	CMPSS4_CTRIPH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS4_CTRIPH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
6	CMPSS4_CTRIPL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS4_CTRIPL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
5	CMPSS3_CTRIPH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS3_CTRIPH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
4	CMPSS3_CTRIPL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS3_CTRIPL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
3	CMPSS2_CTRIPH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS2_CTRIPH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
2	CMPSS2_CTRIPL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS2_CTRIPL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
1	CMPSS1_CTRIPH	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS1_CTRIPH bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
0	CMPSS1_CTRIPL	R-0/W1S	0h	Writing 1 to this bit clears the CMPSS1_CTRIPL bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

### 9.3.2.6 XBARCLR2 Register (Offset = Ah) [reset = 0h]

XBARCLR2 is shown in [Figure 9-11](#) and described in [Table 9-13](#).

Return to the [Summary Table](#).

This register is used to clear the flag(s) in the XBARFLG2 register.

1: Clears the corresponding bit in the XBARFLG2 register.

0: Writing 0 has no effect

**Figure 9-11. XBARCLR2 Register**

31	30	29	28	27	26	25	24
ADCCEVT1	ADCBEVT4	ADCBEVT3	ADCBEVT2	ADCBEVT1	ADCAEVT4	ADCAEVT3	ADCAEVT2
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
23	22	21	20	19	18	17	16
ADCAEVT1	EXTSYNCOUT	ECAP6_OUT	ECAP5_OUT	ECAP4_OUT	ECAP3_OUT	ECAP2_OUT	ECAP1_OUT
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
15	14	13	12	11	10	9	8
INPUT14	INPUT13	INPUT12	INPUT11	INPUT10	INPUT9	INPUT8	INPUT7
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
ADCSOCB	ADCSOCA	INPUT6	INPUT5	INPUT4	INPUT3	INPUT2	INPUT1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 9-13. XBARCLR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	ADCCEVT1	R-0/W1S	0h	Writing 1 to this bit clears the ADCCEVT1 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
30	ADCBEVT4	R-0/W1S	0h	Writing 1 to this bit clears the ADCBEVT4 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
29	ADCBEVT3	R-0/W1S	0h	Writing 1 to this bit clears the ADCBEVT3 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
28	ADCBEVT2	R-0/W1S	0h	Writing 1 to this bit clears the ADCBEVT2 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
27	ADCBEVT1	R-0/W1S	0h	Writing 1 to this bit clears the ADCBEVT1 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
26	ADCAEVT4	R-0/W1S	0h	Writing 1 to this bit clears the ADCAEVT4 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
25	ADCAEVT3	R-0/W1S	0h	Writing 1 to this bit clears the ADCAEVT3 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

**Table 9-13. XBARCLR2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
24	ADCAEVT2	R-0/W1S	0h	Writing 1 to this bit clears the ADCAEVT2 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
23	ADCAEVT1	R-0/W1S	0h	Writing 1 to this bit clears the ADCAEVT1 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
22	EXTSYNCOUT	R-0/W1S	0h	Writing 1 to this bit clears the EXTSYNCOUT bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
21	ECAP6_OUT	R-0/W1S	0h	Writing 1 to this bit clears the ECAP6_OUT bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
20	ECAP5_OUT	R-0/W1S	0h	Writing 1 to this bit clears the ECAP5_OUT bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
19	ECAP4_OUT	R-0/W1S	0h	Writing 1 to this bit clears the ECAP4_OUT bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
18	ECAP3_OUT	R-0/W1S	0h	Writing 1 to this bit clears the ECAP3_OUT bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
17	ECAP2_OUT	R-0/W1S	0h	Writing 1 to this bit clears the ECAP2_OUT bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
16	ECAP1_OUT	R-0/W1S	0h	Writing 1 to this bit clears the ECAP1_OUT bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
15	INPUT14	R-0/W1S	0h	Writing 1 to this bit clears the INPUT14 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
14	INPUT13	R-0/W1S	0h	Writing 1 to this bit clears the INPUT13 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
13	INPUT12	R-0/W1S	0h	Writing 1 to this bit clears the INPUT12 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
12	INPUT11	R-0/W1S	0h	Writing 1 to this bit clears the INPUT11 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

**Table 9-13. XBARCLR2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	INPUT10	R-0/W1S	0h	Writing 1 to this bit clears the INPUT10 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
10	INPUT9	R-0/W1S	0h	Writing 1 to this bit clears the INPUT9 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
9	INPUT8	R-0/W1S	0h	Writing 1 to this bit clears the INPUT8 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
8	INPUT7	R-0/W1S	0h	Writing 1 to this bit clears the INPUT7 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
7	ADCSOCB	R-0/W1S	0h	Writing 1 to this bit clears the ADCSOCB bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
6	ADCSOCA	R-0/W1S	0h	Writing 1 to this bit clears the ADCSOCA bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
5	INPUT6	R-0/W1S	0h	Writing 1 to this bit clears the INPUT6 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
4	INPUT5	R-0/W1S	0h	Writing 1 to this bit clears the INPUT5 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
3	INPUT4	R-0/W1S	0h	Writing 1 to this bit clears the INPUT4 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
2	INPUT3	R-0/W1S	0h	Writing 1 to this bit clears the INPUT3 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
1	INPUT2	R-0/W1S	0h	Writing 1 to this bit clears the INPUT2 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
0	INPUT1	R-0/W1S	0h	Writing 1 to this bit clears the INPUT1 bit in the XBARFLG2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

### 9.3.2.7 XBARCLR3 Register (Offset = Ch) [reset = 0h]

XBARCLR3 is shown in [Figure 9-12](#) and described in [Table 9-14](#).

Return to the [Summary Table](#).

This register is used to clear the flag(s) in the XBARFLG3 register.

1: Clears the corresponding bit in the XBARFLG3 register.

0: Writing 0 has no effect

**Figure 9-12. XBARCLR3 Register**

31		30		29		28		27		26		25		24	
SD1FLT4_DRINT	SD1FLT4_COMPZ	SD1FLT3_DRINT	SD1FLT3_COMPZ	SD1FLT2_DRINT	SD1FLT2_COMPZ	SD1FLT1_DRINT	SD1FLT1_COMPZ								
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h								
23		22		21		20		19		18		17		16	
ECAP7_OUT	SD2FLT4_COMPH	SD2FLT4_COMPL	SD2FLT3_COMPH	SD2FLT3_COMPL	SD2FLT2_COMPH	SD2FLT2_COMPL	SD2FLT1_COMPH								
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h								
15		14		13		12		11		10		9		8	
SD2FLT1_COMPL	SD1FLT4_COMPH	SD1FLT4_COMPL	SD1FLT3_COMPH	SD1FLT3_COMPL	SD1FLT2_COMPH	SD1FLT2_COMPL	SD1FLT1_COMPH								
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h								
7		6		5		4		3		2		1		0	
SD1FLT1_COMPL										ADCCEVT4	ADCCEVT3	ADCCEVT2			
R-0/W1S-0h										R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h			

**Table 9-14. XBARCLR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	SD1FLT4_DRINT	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT4_DRINT bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
30	SD1FLT4_COMPZ	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT4_COMPZ bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
29	SD1FLT3_DRINT	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT3_DRINT bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
28	SD1FLT3_COMPZ	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT3_COMPZ bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
27	SD1FLT2_DRINT	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT2_DRINT bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
26	SD1FLT2_COMPZ	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT2_COMPZ bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

**Table 9-14. XBARCLR3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25	SD1FLT1_DRINT	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT1_DRINT bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
24	SD1FLT1_COMPZ	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT1_COMPZ bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
23	ECAP7_OUT	R-0/W1S	0h	Writing 1 to this bit clears the ECAP7_OUT bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
22	SD2FLT4_COMPH	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT4_COMPH bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
21	SD2FLT4_COMPL	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT4_COMPL bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
20	SD2FLT3_COMPH	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT3_COMPH bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
19	SD2FLT3_COMPL	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT3_COMPL bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
18	SD2FLT2_COMPH	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT2_COMPH bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
17	SD2FLT2_COMPL	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT2_COMPL bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
16	SD2FLT1_COMPH	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT1_COMPH bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
15	SD2FLT1_COMPL	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT1_COMPL bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
14	SD1FLT4_COMPH	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT4_COMPH bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
13	SD1FLT4_COMPL	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT4_COMPL bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

**Table 9-14. XBARCLR3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	SD1FLT3_COMPH	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT3_COMPH bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
11	SD1FLT3_COMPL	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT3_COMPL bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
10	SD1FLT2_COMPH	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT2_COMPH bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
9	SD1FLT2_COMPL	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT2_COMPL bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
8	SD1FLT1_COMPH	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT1_COMPH bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
7	SD1FLT1_COMPL	R-0/W1S	0h	Writing 1 to this bit clears the SD1FLT1_COMPL bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
6	RESERVED	R-0/W1S	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	ADCCEVT4	R-0/W1S	0h	Writing 1 to this bit clears the ADCCEVT4 bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
1	ADCCEVT3	R-0/W1S	0h	Writing 1 to this bit clears the ADCCEVT3 bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
0	ADCCEVT2	R-0/W1S	0h	Writing 1 to this bit clears the ADCCEVT2 bit in the XBARFLG3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn



### 9.3.2.8 XBARCLR4 Register (Offset = Eh) [reset = 0h]

XBARCLR4 is shown in [Figure 9-13](#) and described in [Table 9-15](#).

Return to the [Summary Table](#).

This register is used to clear the flag(s) in the XBARFLG4 register.

1: Clears the corresponding bit in the XBARFLG4 register.

0: Writing 0 has no effect

**Figure 9-13. XBARCLR4 Register**

31		30		29		28		27		26		25		24	
CLAHALT															
R-0h															
23		22		21		20		19		18		17		16	
CLB4_OUT5	CLB4_OUT4	CLB3_OUT5	CLB3_OUT4	CLB2_OUT5	CLB2_OUT4	CLB1_OUT5	CLB1_OUT4								
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h								
15		14		13		12		11		10		9		8	
CLB8_5_1	CLB8_4_1	CLB7_5_1	CLB7_4_1												
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h												
7		6		5		4		3		2		1		0	
SD2FLT4_DRI NT	SD2FLT4_CO MPZ	SD2FLT3_DRI NT	SD2FLT3_CO MPZ	SD2FLT2_DRI NT	SD2FLT2_CO MPZ	SD2FLT1_DRI NT	SD2FLT1_CO MPZ								
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h								

**Table 9-15. XBARCLR4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	CLAHALT	R	0h	Writing 1 to this bit clears the CLAHALT bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
30	RESERVED	R-0/W1S	0h	Reserved
29	RESERVED	R-0/W1S	0h	Reserved
28	RESERVED	R-0/W1S	0h	Reserved
27	RESERVED	R-0/W1S	0h	Reserved
26	RESERVED	R-0/W1S	0h	Reserved
25	RESERVED	R-0/W1S	0h	Reserved
24	RESERVED	R-0/W1S	0h	Reserved
23	CLB4_OUT5	R-0/W1S	0h	Writing 1 to this bit clears the CLB4_OUT5 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
22	CLB4_OUT4	R-0/W1S	0h	Writing 1 to this bit clears the CLB4_OUT4 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
21	CLB3_OUT5	R-0/W1S	0h	Writing 1 to this bit clears the CLB3_OUT5 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

**Table 9-15. XBARCLR4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	CLB3_OUT4	R-0/W1S	0h	Writing 1 to this bit clears the CLB3_OUT4 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
19	CLB2_OUT5	R-0/W1S	0h	Writing 1 to this bit clears the CLB2_OUT5 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
18	CLB2_OUT4	R-0/W1S	0h	Writing 1 to this bit clears the CLB2_OUT4 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
17	CLB1_OUT5	R-0/W1S	0h	Writing 1 to this bit clears the CLB1_OUT5 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
16	CLB1_OUT4	R-0/W1S	0h	Writing 1 to this bit clears the CLB1_OUT4 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
15	CLB8_5_1	R-0/W1S	0h	Writing 1 to this bit clears the CLB8_5_1 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
14	CLB8_4_1	R-0/W1S	0h	Writing 1 to this bit clears the CLB8_4_1 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
13	CLB7_5_1	R-0/W1S	0h	Writing 1 to this bit clears the CLB7_5_1 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
12	CLB7_4_1	R-0/W1S	0h	Writing 1 to this bit clears the CLB7_4_1 bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	SD2FLT4_DRINT	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT4_DRINT bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
6	SD2FLT4_COMPZ	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT4_COMPZ bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

**Table 9-15. XBARCLR4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	SD2FLT3_DRINT	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT3_DRINT bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
4	SD2FLT3_COMPZ	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT3_COMPZ bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
3	SD2FLT2_DRINT	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT2_DRINT bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
2	SD2FLT2_COMPZ	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT2_COMPZ bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
1	SD2FLT1_DRINT	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT1_DRINT bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
0	SD2FLT1_COMPZ	R-0/W1S	0h	Writing 1 to this bit clears the SD2FLT1_COMPZ bit in the XBARFLG4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

### 9.3.3 INPUT\_XBAR\_REGS Registers

Table 9-16 lists the INPUT\_XBAR\_REGS registers. All register offset addresses not listed in Table 9-16 should be considered as reserved locations and the register contents should not be modified.

**Table 9-16. INPUT\_XBAR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	INPUT1SELECT	INPUT1 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
1h	INPUT2SELECT	INPUT2 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
2h	INPUT3SELECT	INPUT3 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
3h	INPUT4SELECT	INPUT4 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
4h	INPUT5SELECT	INPUT5 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
5h	INPUT6SELECT	INPUT6 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
6h	INPUT7SELECT	INPUT7 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
7h	INPUT8SELECT	INPUT8 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
8h	INPUT9SELECT	INPUT9 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
9h	INPUT10SELECT	INPUT10 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Ah	INPUT11SELECT	INPUT11 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Bh	INPUT12SELECT	INPUT12 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Ch	INPUT13SELECT	INPUT13 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Dh	INPUT14SELECT	INPUT14 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Eh	INPUT15SELECT	INPUT15 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Fh	INPUT16SELECT	INPUT16 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
1Eh	INPUTSELECTLOCK	Input Select Lock Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 9-17 shows the codes that are used for access types in this section.

**Table 9-17. INPUT\_XBAR\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
WOnce	W Once	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 9.3.3.1 INPUT1SELECT Register (Offset = 0h) [reset = FFFEh]

INPUT1SELECT is shown in [Figure 9-14](#) and described in [Table 9-18](#).

Return to the [Summary Table](#).

INPUT1 Input Select Register (GPIO0 to x)

**Figure 9-14. INPUT1SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-18. INPUT1SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT1 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

### 9.3.3.2 INPUT2SELECT Register (Offset = 1h) [reset = FFFEh]

INPUT2SELECT is shown in [Figure 9-15](#) and described in [Table 9-19](#).

Return to the [Summary Table](#).

INPUT2 Input Select Register (GPIO0 to x)

**Figure 9-15. INPUT2SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-19. INPUT2SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT2 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

### 9.3.3.3 INPUT3SELECT Register (Offset = 2h) [reset = FFFEh]

INPUT3SELECT is shown in [Figure 9-16](#) and described in [Table 9-20](#).

Return to the [Summary Table](#).

INPUT3 Input Select Register (GPIO0 to x)

**Figure 9-16. INPUT3SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-20. INPUT3SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT3 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

### 9.3.3.4 INPUT4SELECT Register (Offset = 3h) [reset = FFFEh]

INPUT4SELECT is shown in [Figure 9-17](#) and described in [Table 9-21](#).

Return to the [Summary Table](#).

INPUT4 Input Select Register (GPIO0 to x)

**Figure 9-17. INPUT4SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-21. INPUT4SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	<p>Select GPIO for INPUT4 signal:</p> <p>0x0 : Select GPIO0</p> <p>0x1 : Select GPIO1</p> <p>0x2 : Select GPIO2</p> <p>...</p> <p>0xFFFFD: '1' will be driven to the destination</p> <p>0xFFFFE: '1' will be driven to the destination</p> <p>0xFFFFF: '0' will be driven to the destination</p> <p>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.</p> <p>Reset type: CPU1.SYSRSn</p>



### 9.3.3.5 INPUT5SELECT Register (Offset = 4h) [reset = FFFEh]

INPUT5SELECT is shown in [Figure 9-18](#) and described in [Table 9-22](#).

Return to the [Summary Table](#).

INPUT5 Input Select Register (GPIO0 to x)

**Figure 9-18. INPUT5SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-22. INPUT5SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT5 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

### 9.3.3.6 INPUT6SELECT Register (Offset = 5h) [reset = FFFEh]

INPUT6SELECT is shown in [Figure 9-19](#) and described in [Table 9-23](#).

Return to the [Summary Table](#).

INPUT6 Input Select Register (GPIO0 to x)

**Figure 9-19. INPUT6SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-23. INPUT6SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT6 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

### 9.3.3.7 INPUT7SELECT Register (Offset = 6h) [reset = FFFEh]

INPUT7SELECT is shown in [Figure 9-20](#) and described in [Table 9-24](#).

Return to the [Summary Table](#).

INPUT7 Input Select Register (GPIO0 to x)

**Figure 9-20. INPUT7SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-24. INPUT7SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT7 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

### 9.3.3.8 INPUT8SELECT Register (Offset = 7h) [reset = FFFEh]

INPUT8SELECT is shown in [Figure 9-21](#) and described in [Table 9-25](#).

Return to the [Summary Table](#).

INPUT8 Input Select Register (GPIO0 to x)

**Figure 9-21. INPUT8SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-25. INPUT8SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT8 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

### 9.3.3.9 INPUT9SELECT Register (Offset = 8h) [reset = FFFEh]

INPUT9SELECT is shown in [Figure 9-22](#) and described in [Table 9-26](#).

Return to the [Summary Table](#).

INPUT9 Input Select Register (GPIO0 to x)

**Figure 9-22. INPUT9SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-26. INPUT9SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT9 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

**9.3.3.10 INPUT10SELECT Register (Offset = 9h) [reset = FFFEh]**

INPUT10SELECT is shown in [Figure 9-23](#) and described in [Table 9-27](#).

Return to the [Summary Table](#).

INPUT10 Input Select Register (GPIO0 to x)

**Figure 9-23. INPUT10SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-27. INPUT10SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT10 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

### 9.3.3.11 INPUT11SELECT Register (Offset = Ah) [reset = FFFEh]

INPUT11SELECT is shown in [Figure 9-24](#) and described in [Table 9-28](#).

Return to the [Summary Table](#).

INPUT11 Input Select Register (GPIO0 to x)

**Figure 9-24. INPUT11SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-28. INPUT11SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT11 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

**9.3.3.12 INPUT12SELECT Register (Offset = Bh) [reset = FFFEh]**

INPUT12SELECT is shown in [Figure 9-25](#) and described in [Table 9-29](#).

Return to the [Summary Table](#).

INPUT12 Input Select Register (GPIO0 to x)

**Figure 9-25. INPUT12SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-29. INPUT12SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT12 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn



### 9.3.3.13 INPUT13SELECT Register (Offset = Ch) [reset = FFFEh]

INPUT13SELECT is shown in [Figure 9-26](#) and described in [Table 9-30](#).

Return to the [Summary Table](#).

INPUT13 Input Select Register (GPIO0 to x)

**Figure 9-26. INPUT13SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-30. INPUT13SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT13 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFD: '1' will be driven to the destination 0xFFFE: '1' will be driven to the destination 0xFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

**9.3.3.14 INPUT14SELECT Register (Offset = Dh) [reset = FFFEh]**

INPUT14SELECT is shown in [Figure 9-27](#) and described in [Table 9-31](#).

Return to the [Summary Table](#).

INPUT14 Input Select Register (GPIO0 to x)

**Figure 9-27. INPUT14SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-31. INPUT14SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT14 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

**9.3.3.15 INPUT15SELECT Register (Offset = Eh) [reset = FFFEh]**

INPUT15SELECT is shown in [Figure 9-28](#) and described in [Table 9-32](#).

Return to the [Summary Table](#).

INPUT15 Input Select Register (GPIO0 to x)

**Figure 9-28. INPUT15SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-32. INPUT15SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT15 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFD: '1' will be driven to the destination 0xFFFE: '1' will be driven to the destination 0xFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

**9.3.3.16 INPUT16SELECT Register (Offset = Fh) [reset = FFFEh]**

INPUT16SELECT is shown in [Figure 9-29](#) and described in [Table 9-33](#).

Return to the [Summary Table](#).

INPUT16 Input Select Register (GPIO0 to x)

**Figure 9-29. INPUT16SELECT Register**

15	14	13	12	11	10	9	8
SELECT							
R/W-FFFEh							
7	6	5	4	3	2	1	0
SELECT							
R/W-FFFEh							

**Table 9-33. INPUT16SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT16 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination  NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.  Reset type: CPU1.SYSRSn

### 9.3.3.17 INPUTSELECTLOCK Register (Offset = 1Eh) [reset = 0h]

INPUTSELECTLOCK is shown in [Figure 9-30](#) and described in [Table 9-34](#).

Return to the [Summary Table](#).

Input Select Lock Register.

Any bit in this register, once set can only be cleared through SYSRSn. Write of 0 to any bit of this register has no effect. Reads to the registers which have LOCK protection are always allowed.

**Figure 9-30. INPUTSELECTLOCK Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
INPUT16SELE CT	INPUT15SELE CT	INPUT14SELE CT	INPUT13SELE CT	INPUT12SELE CT	INPUT11SELE CT	INPUT10SELE CT	INPUT9SELEC T
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h
7	6	5	4	3	2	1	0
INPUT8SELEC T	INPUT7SELEC T	INPUT6SELEC T	INPUT5SELEC T	INPUT4SELEC T	INPUT3SELEC T	INPUT2SELEC T	INPUT1SELEC T
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 9-34. INPUTSELECTLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15	INPUT16SELECT	R/WOnce	0h	Lock bit for INPUT16SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
14	INPUT15SELECT	R/WOnce	0h	Lock bit for INPUT15SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
13	INPUT14SELECT	R/WOnce	0h	Lock bit for INPUT14SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
12	INPUT13SELECT	R/WOnce	0h	Lock bit for INPUT13SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
11	INPUT12SELECT	R/WOnce	0h	Lock bit for INPUT12SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
10	INPUT11SELECT	R/WOnce	0h	Lock bit for INPUT11SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn

**Table 9-34. INPUTSELECTLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	INPUT10SELECT	R/WOnce	0h	Lock bit for INPUT10SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
8	INPUT9SELECT	R/WOnce	0h	Lock bit for INPUT9SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
7	INPUT8SELECT	R/WOnce	0h	Lock bit for INPUT8SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
6	INPUT7SELECT	R/WOnce	0h	Lock bit for INPUT7SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
5	INPUT6SELECT	R/WOnce	0h	Lock bit for INPUT6SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
4	INPUT5SELECT	R/WOnce	0h	Lock bit for INPUT5SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
3	INPUT4SELECT	R/WOnce	0h	Lock bit for INPUT4SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
2	INPUT3SELECT	R/WOnce	0h	Lock bit for INPUT3SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
1	INPUT2SELECT	R/WOnce	0h	Lock bit for INPUT2SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
0	INPUT1SELECT	R/WOnce	0h	Lock bit for INPUT1SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn

### 9.3.4 OUTPUT\_XBAR\_REGS Registers

Table 9-35 lists the OUTPUT\_XBAR\_REGS registers. All register offset addresses not listed in Table 9-35 should be considered as reserved locations and the register contents should not be modified.

**Table 9-35. OUTPUT\_XBAR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	OUTPUT1MUX0TO15CFG	Output X-BAR Mux Configuration for Output 1	EALLOW	<a href="#">Go</a>
2h	OUTPUT1MUX16TO31CFG	Output X-BAR Mux Configuration for Output 1	EALLOW	<a href="#">Go</a>
4h	OUTPUT2MUX0TO15CFG	Output X-BAR Mux Configuration for Output 2	EALLOW	<a href="#">Go</a>
6h	OUTPUT2MUX16TO31CFG	Output X-BAR Mux Configuration for Output 2	EALLOW	<a href="#">Go</a>
8h	OUTPUT3MUX0TO15CFG	Output X-BAR Mux Configuration for Output 3	EALLOW	<a href="#">Go</a>
Ah	OUTPUT3MUX16TO31CFG	Output X-BAR Mux Configuration for Output 3	EALLOW	<a href="#">Go</a>
Ch	OUTPUT4MUX0TO15CFG	Output X-BAR Mux Configuration for Output 4	EALLOW	<a href="#">Go</a>
Eh	OUTPUT4MUX16TO31CFG	Output X-BAR Mux Configuration for Output 4	EALLOW	<a href="#">Go</a>
10h	OUTPUT5MUX0TO15CFG	Output X-BAR Mux Configuration for Output 5	EALLOW	<a href="#">Go</a>
12h	OUTPUT5MUX16TO31CFG	Output X-BAR Mux Configuration for Output 5	EALLOW	<a href="#">Go</a>
14h	OUTPUT6MUX0TO15CFG	Output X-BAR Mux Configuration for Output 6	EALLOW	<a href="#">Go</a>
16h	OUTPUT6MUX16TO31CFG	Output X-BAR Mux Configuration for Output 6	EALLOW	<a href="#">Go</a>
18h	OUTPUT7MUX0TO15CFG	Output X-BAR Mux Configuration for Output 7	EALLOW	<a href="#">Go</a>
1Ah	OUTPUT7MUX16TO31CFG	Output X-BAR Mux Configuration for Output 7	EALLOW	<a href="#">Go</a>
1Ch	OUTPUT8MUX0TO15CFG	Output X-BAR Mux Configuration for Output 8	EALLOW	<a href="#">Go</a>
1Eh	OUTPUT8MUX16TO31CFG	Output X-BAR Mux Configuration for Output 8	EALLOW	<a href="#">Go</a>
20h	OUTPUT1MUXENABLE	Output X-BAR Mux Enable for Output 1	EALLOW	<a href="#">Go</a>
22h	OUTPUT2MUXENABLE	Output X-BAR Mux Enable for Output 2	EALLOW	<a href="#">Go</a>
24h	OUTPUT3MUXENABLE	Output X-BAR Mux Enable for Output 3	EALLOW	<a href="#">Go</a>
26h	OUTPUT4MUXENABLE	Output X-BAR Mux Enable for Output 4	EALLOW	<a href="#">Go</a>
28h	OUTPUT5MUXENABLE	Output X-BAR Mux Enable for Output 5	EALLOW	<a href="#">Go</a>
2Ah	OUTPUT6MUXENABLE	Output X-BAR Mux Enable for Output 6	EALLOW	<a href="#">Go</a>
2Ch	OUTPUT7MUXENABLE	Output X-BAR Mux Enable for Output 7	EALLOW	<a href="#">Go</a>
2Eh	OUTPUT8MUXENABLE	Output X-BAR Mux Enable for Output 8	EALLOW	<a href="#">Go</a>
30h	OUTPUTLATCH	Output X-BAR Output Latch		<a href="#">Go</a>
32h	OUTPUTLATCHCLR	Output X-BAR Output Latch Clear		<a href="#">Go</a>
34h	OUTPUTLATCHFRC	Output X-BAR Output Latch Clear		<a href="#">Go</a>
36h	OUTPUTLATCHENABLE	Output X-BAR Output Latch Enable	EALLOW	<a href="#">Go</a>
38h	OUTPUTINV	Output X-BAR Output Inversion	EALLOW	<a href="#">Go</a>
3Eh	OUTPUTLOCK	Output X-BAR Configuration Lock register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 9-36 shows the codes that are used for access types in this section.

**Table 9-36. OUTPUT\_XBAR\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set

**Table 9-36. OUTPUT\_XBAR\_REGS Access Type Codes (continued)**

Access Type	Code	Description
WSoOnce	W SoOnce	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.



### 9.3.4.1 OUTPUT1MUX0TO15CFG Register (Offset = 0h) [reset = 0h]

OUTPUT1MUX0TO15CFG is shown in [Figure 9-31](#) and described in [Table 9-37](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 1

**Figure 9-31. OUTPUT1MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-37. OUTPUT1MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT1 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT1 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT1 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT1 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-37. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for OUTPUT1 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT1 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for OUTPUT1 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT1 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT1 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT1 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-37. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for OUTPUT1 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT1 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT1 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT1 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for OUTPUT1 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT1 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.2 OUTPUT1MUX16TO31CFG Register (Offset = 2h) [reset = 0h]

OUTPUT1MUX16TO31CFG is shown in [Figure 9-32](#) and described in [Table 9-38](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 1

**Figure 9-32. OUTPUT1MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-38. OUTPUT1MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT1 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT1 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT1 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT1 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-38. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for OUTPUT1 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT1 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for OUTPUT1 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT1 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT1 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT1 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-38. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for OUTPUT1 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT1 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT1 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT1 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for OUTPUT1 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT1 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.3 OUTPUT2MUX0TO15CFG Register (Offset = 4h) [reset = 0h]

OUTPUT2MUX0TO15CFG is shown in [Figure 9-33](#) and described in [Table 9-39](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 2

**Figure 9-33. OUTPUT2MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-39. OUTPUT2MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT2 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT2 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT2 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT2 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-39. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for OUTPUT2 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT2 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for OUTPUT2 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT2 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT2 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT2 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



**Table 9-39. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for OUTPUT2 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT2 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT2 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT2 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for OUTPUT2 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT2 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**9.3.4.4 OUTPUT2MUX16TO31CFG Register (Offset = 6h) [reset = 0h]**

OUTPUT2MUX16TO31CFG is shown in [Figure 9-34](#) and described in [Table 9-40](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 2

**Figure 9-34. OUTPUT2MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-40. OUTPUT2MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT2 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT2 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT2 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT2 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-40. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for OUTPUT2 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT2 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for OUTPUT2 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT2 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT2 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT2 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-40. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for OUTPUT2 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT2 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT2 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT2 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for OUTPUT2 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT2 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.5 OUTPUT3MUX0TO15CFG Register (Offset = 8h) [reset = 0h]

OUTPUT3MUX0TO15CFG is shown in [Figure 9-35](#) and described in [Table 9-41](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 3

**Figure 9-35. OUTPUT3MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-41. OUTPUT3MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT3 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT3 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT3 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT3 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-41. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Select Bits for OUTPUT3 Mux11:</p> <p>00 : Select .0 input for Mux11  01 : Select .1 input for Mux11  10 : Select .2 input for Mux11  11 : Select .3 input for Mux11</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Select Bits for OUTPUT3 Mux10:</p> <p>00 : Select .0 input for Mux10  01 : Select .1 input for Mux10  10 : Select .2 input for Mux10  11 : Select .3 input for Mux10</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Select Bits for OUTPUT3 Mux9:</p> <p>00 : Select .0 input for Mux9  01 : Select .1 input for Mux9  10 : Select .2 input for Mux9  11 : Select .3 input for Mux9</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Select Bits for OUTPUT3 Mux8:</p> <p>00 : Select .0 input for Mux8  01 : Select .1 input for Mux8  10 : Select .2 input for Mux8  11 : Select .3 input for Mux8</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Select Bits for OUTPUT3 Mux7:</p> <p>00 : Select .0 input for Mux7  01 : Select .1 input for Mux7  10 : Select .2 input for Mux7  11 : Select .3 input for Mux7</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
13-12	MUX6	R/W	0h	<p>Select Bits for OUTPUT3 Mux6:</p> <p>00 : Select .0 input for Mux6  01 : Select .1 input for Mux6  10 : Select .2 input for Mux6  11 : Select .3 input for Mux6</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-41. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for OUTPUT3 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT3 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT3 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT3 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for OUTPUT3 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT3 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.6 OUTPUT3MUX16TO31CFG Register (Offset = Ah) [reset = 0h]

OUTPUT3MUX16TO31CFG is shown in [Figure 9-36](#) and described in [Table 9-42](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 3

**Figure 9-36. OUTPUT3MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-42. OUTPUT3MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT3 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT3 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT3 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT3 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



**Table 9-42. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for OUTPUT3 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT3 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for OUTPUT3 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT3 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT3 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT3 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-42. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for OUTPUT3 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT3 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT3 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT3 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for OUTPUT3 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT3 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.7 OUTPUT4MUX0TO15CFG Register (Offset = Ch) [reset = 0h]

OUTPUT4MUX0TO15CFG is shown in [Figure 9-37](#) and described in [Table 9-43](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 4

**Figure 9-37. OUTPUT4MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-43. OUTPUT4MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT4 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT4 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT4 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT4 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-43. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for OUTPUT4 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT4 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for OUTPUT4 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT4 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT4 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT4 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-43. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for OUTPUT4 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT4 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT4 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT4 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for OUTPUT4 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT4 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**9.3.4.8 OUTPUT4MUX16TO31CFG Register (Offset = Eh) [reset = 0h]**

OUTPUT4MUX16TO31CFG is shown in [Figure 9-38](#) and described in [Table 9-44](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 4

**Figure 9-38. OUTPUT4MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-44. OUTPUT4MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT4 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT4 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT4 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT4 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-44. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for OUTPUT4 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT4 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for OUTPUT4 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT4 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT4 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT4 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-44. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for OUTPUT4 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT4 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT4 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT4 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for OUTPUT4 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT4 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



### 9.3.4.9 OUTPUT5MUX0TO15CFG Register (Offset = 10h) [reset = 0h]

OUTPUT5MUX0TO15CFG is shown in [Figure 9-39](#) and described in [Table 9-45](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 5

**Figure 9-39. OUTPUT5MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-45. OUTPUT5MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT5 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT5 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT5 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT5 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-45. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Select Bits for OUTPUT5 Mux11:</p> <p>00 : Select .0 input for Mux11  01 : Select .1 input for Mux11  10 : Select .2 input for Mux11  11 : Select .3 input for Mux11</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Select Bits for OUTPUT5 Mux10:</p> <p>00 : Select .0 input for Mux10  01 : Select .1 input for Mux10  10 : Select .2 input for Mux10  11 : Select .3 input for Mux10</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Select Bits for OUTPUT5 Mux9:</p> <p>00 : Select .0 input for Mux9  01 : Select .1 input for Mux9  10 : Select .2 input for Mux9  11 : Select .3 input for Mux9</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Select Bits for OUTPUT5 Mux8:</p> <p>00 : Select .0 input for Mux8  01 : Select .1 input for Mux8  10 : Select .2 input for Mux8  11 : Select .3 input for Mux8</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Select Bits for OUTPUT5 Mux7:</p> <p>00 : Select .0 input for Mux7  01 : Select .1 input for Mux7  10 : Select .2 input for Mux7  11 : Select .3 input for Mux7</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
13-12	MUX6	R/W	0h	<p>Select Bits for OUTPUT5 Mux6:</p> <p>00 : Select .0 input for Mux6  01 : Select .1 input for Mux6  10 : Select .2 input for Mux6  11 : Select .3 input for Mux6</p> <p>Refer to the Output X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-45. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	<p>Select Bits for OUTPUT5 Mux5:</p> <p>00 : Select .0 input for Mux5            01 : Select .1 input for Mux5            10 : Select .2 input for Mux5            11 : Select .3 input for Mux5</p> <p>Refer to the Output X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Select Bits for OUTPUT5 Mux4:</p> <p>00 : Select .0 input for Mux4            01 : Select .1 input for Mux4            10 : Select .2 input for Mux4            11 : Select .3 input for Mux4</p> <p>Refer to the Output X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Select Bits for OUTPUT5 Mux3:</p> <p>00 : Select .0 input for Mux3            01 : Select .1 input for Mux3            10 : Select .2 input for Mux3            11 : Select .3 input for Mux3</p> <p>Refer to the Output X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Select Bits for OUTPUT5 Mux2:</p> <p>00 : Select .0 input for Mux2            01 : Select .1 input for Mux2            10 : Select .2 input for Mux2            11 : Select .3 input for Mux2</p> <p>Refer to the Output X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
3-2	MUX1	R/W	0h	<p>Select Bits for OUTPUT5 Mux1:</p> <p>00 : Select .0 input for Mux1            01 : Select .1 input for Mux1            10 : Select .2 input for Mux1            11 : Select .3 input for Mux1</p> <p>Refer to the Output X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Select Bits for OUTPUT5 Mux0:</p> <p>00 : Select .0 input for Mux0            01 : Select .1 input for Mux0            10 : Select .2 input for Mux0            11 : Select .3 input for Mux0</p> <p>Refer to the Output X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

### 9.3.4.10 OUTPUT5MUX16TO31CFG Register (Offset = 12h) [reset = 0h]

OUTPUT5MUX16TO31CFG is shown in [Figure 9-40](#) and described in [Table 9-46](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 5

**Figure 9-40. OUTPUT5MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-46. OUTPUT5MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT5 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT5 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT5 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT5 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-46. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for OUTPUT5 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT5 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for OUTPUT5 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT5 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT5 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT5 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-46. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for OUTPUT5 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT5 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT5 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT5 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for OUTPUT5 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT5 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.11 OUTPUT6MUX0TO15CFG Register (Offset = 14h) [reset = 0h]

OUTPUT6MUX0TO15CFG is shown in [Figure 9-41](#) and described in [Table 9-47](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 6

**Figure 9-41. OUTPUT6MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-47. OUTPUT6MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT6 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT6 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT6 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT6 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-47. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for OUTPUT6 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT6 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for OUTPUT6 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT6 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT6 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT6 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



**Table 9-47. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for OUTPUT6 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT6 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT6 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT6 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for OUTPUT6 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT6 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**9.3.4.12 OUTPUT6MUX16TO31CFG Register (Offset = 16h) [reset = 0h]**

OUTPUT6MUX16TO31CFG is shown in [Figure 9-42](#) and described in [Table 9-48](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 6

**Figure 9-42. OUTPUT6MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-48. OUTPUT6MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT6 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT6 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT6 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT6 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-48. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for OUTPUT6 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT6 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for OUTPUT6 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT6 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT6 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT6 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-48. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for OUTPUT6 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT6 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT6 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT6 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for OUTPUT6 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT6 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.13 OUTPUT7MUX0TO15CFG Register (Offset = 18h) [reset = 0h]

OUTPUT7MUX0TO15CFG is shown in [Figure 9-43](#) and described in [Table 9-49](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 7

**Figure 9-43. OUTPUT7MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-49. OUTPUT7MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT7 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT7 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT7 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT7 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-49. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for OUTPUT7 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT7 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for OUTPUT7 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT7 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT7 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT7 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-49. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for OUTPUT7 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT7 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT7 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT7 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for OUTPUT7 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT7 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**9.3.4.14 OUTPUT7MUX16TO31CFG Register (Offset = 1Ah) [reset = 0h]**

OUTPUT7MUX16TO31CFG is shown in [Figure 9-44](#) and described in [Table 9-50](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 7

**Figure 9-44. OUTPUT7MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-50. OUTPUT7MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT7 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT7 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT7 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT7 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



**Table 9-50. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for OUTPUT7 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT7 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for OUTPUT7 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT7 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT7 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT7 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-50. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for OUTPUT7 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT7 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT7 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT7 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for OUTPUT7 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT7 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.15 OUTPUT8MUX0TO15CFG Register (Offset = 1Ch) [reset = 0h]

OUTPUT8MUX0TO15CFG is shown in [Figure 9-45](#) and described in [Table 9-51](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 8

**Figure 9-45. OUTPUT8MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-51. OUTPUT8MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT8 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT8 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT8 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT8 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-51. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for OUTPUT8 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT8 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for OUTPUT8 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT8 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT8 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT8 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-51. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for OUTPUT8 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT8 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT8 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT8 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for OUTPUT8 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT8 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.16 OUTPUT8MUX16TO31CFG Register (Offset = 1Eh) [reset = 0h]

OUTPUT8MUX16TO31CFG is shown in [Figure 9-46](#) and described in [Table 9-52](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 8

**Figure 9-46. OUTPUT8MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-52. OUTPUT8MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT8 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT8 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT8 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT8 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-52. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for OUTPUT8 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT8 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for OUTPUT8 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT8 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT8 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT8 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-52. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for OUTPUT8 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT8 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT8 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT8 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for OUTPUT8 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT8 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



### 9.3.4.17 OUTPUT1MUXENABLE Register (Offset = 20h) [reset = 0h]

OUTPUT1MUXENABLE is shown in [Figure 9-47](#) and described in [Table 9-53](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 1

**Figure 9-47. OUTPUT1MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
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
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
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
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
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 9-53. OUTPUT1MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT1 of OUTPUT-XBAR  0: Respective output of Mux31 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT1 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT1 of OUTPUT-XBAR  0: Respective output of Mux30 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT1 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT1 of OUTPUT-XBAR  0: Respective output of Mux29 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT1 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT1 of OUTPUT-XBAR  0: Respective output of Mux28 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT1 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-53. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-53. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-53. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-53. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive OUTPUT1 of OUTPUT-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the OUTPUT1 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.4.18 OUTPUT2MUXENABLE Register (Offset = 22h) [reset = 0h]

OUTPUT2MUXENABLE is shown in [Figure 9-48](#) and described in [Table 9-54](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 2

**Figure 9-48. OUTPUT2MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-54. OUTPUT2MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux31 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux30 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux29 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux28 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-54. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-54. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>



**Table 9-54. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-54. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive OUTPUT2 of OUTPUT-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the OUTPUT2 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.4.19 OUTPUT3MUXENABLE Register (Offset = 24h) [reset = 0h]

OUTPUT3MUXENABLE is shown in [Figure 9-49](#) and described in [Table 9-55](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 3

**Figure 9-49. OUTPUT3MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
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
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
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
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
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 9-55. OUTPUT3MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT3 of OUTPUT-XBAR  0: Respective output of Mux31 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT3 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT3 of OUTPUT-XBAR  0: Respective output of Mux30 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT3 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT3 of OUTPUT-XBAR  0: Respective output of Mux29 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT3 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT3 of OUTPUT-XBAR  0: Respective output of Mux28 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT3 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-55. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-55. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-55. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-55. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive OUTPUT3 of OUTPUT-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the OUTPUT3 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.4.20 OUTPUT4MUXENABLE Register (Offset = 26h) [reset = 0h]

OUTPUT4MUXENABLE is shown in [Figure 9-50](#) and described in [Table 9-56](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 4

**Figure 9-50. OUTPUT4MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-56. OUTPUT4MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT4 of OUTPUT-XBAR  0: Respective output of Mux31 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT4 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT4 of OUTPUT-XBAR  0: Respective output of Mux30 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT4 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT4 of OUTPUT-XBAR  0: Respective output of Mux29 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT4 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT4 of OUTPUT-XBAR  0: Respective output of Mux28 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT4 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



**Table 9-56. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-56. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-56. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-56. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive OUTPUT4 of OUTPUT-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the OUTPUT4 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.4.21 OUTPUT5MUXENABLE Register (Offset = 28h) [reset = 0h]

OUTPUT5MUXENABLE is shown in [Figure 9-51](#) and described in [Table 9-57](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 5

**Figure 9-51. OUTPUT5MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
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
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
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
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
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 9-57. OUTPUT5MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT5 of OUTPUT-XBAR  0: Respective output of Mux31 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT5 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT5 of OUTPUT-XBAR  0: Respective output of Mux30 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT5 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT5 of OUTPUT-XBAR  0: Respective output of Mux29 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT5 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT5 of OUTPUT-XBAR  0: Respective output of Mux28 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT5 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-57. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-57. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-57. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>



**Table 9-57. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive OUTPUT5 of OUTPUT-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the OUTPUT5 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.4.22 OUTPUT6MUXENABLE Register (Offset = 2Ah) [reset = 0h]

OUTPUT6MUXENABLE is shown in [Figure 9-52](#) and described in [Table 9-58](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 6

**Figure 9-52. OUTPUT6MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-58. OUTPUT6MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT6 of OUTPUT-XBAR  0: Respective output of Mux31 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT6 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT6 of OUTPUT-XBAR  0: Respective output of Mux30 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT6 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT6 of OUTPUT-XBAR  0: Respective output of Mux29 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT6 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT6 of OUTPUT-XBAR  0: Respective output of Mux28 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT6 of OUTPUT-XBAR  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-58. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-58. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-58. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-58. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive OUTPUT6 of OUTPUT-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the OUTPUT6 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.4.23 OUTPUT7MUXENABLE Register (Offset = 2Ch) [reset = 0h]

OUTPUT7MUXENABLE is shown in [Figure 9-53](#) and described in [Table 9-59](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 7

**Figure 9-53. OUTPUT7MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
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
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
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
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
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 9-59. OUTPUT7MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of Mux31 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux31 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux31 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of Mux30 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux30 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux30 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of Mux29 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux29 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux29 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of Mux28 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux28 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux28 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-59. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>



**Table 9-59. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-59. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-59. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive OUTPUT7 of OUTPUT-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the OUTPUT7 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.4.24 OUTPUT8MUXENABLE Register (Offset = 2Eh) [reset = 0h]

OUTPUT8MUXENABLE is shown in [Figure 9-54](#) and described in [Table 9-60](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 8

**Figure 9-54. OUTPUT8MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-60. OUTPUT8MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of Mux31 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux31 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux31 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of Mux30 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux30 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux30 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of Mux29 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux29 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux29 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of Mux28 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux28 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux28 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-60. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-60. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-60. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-60. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive OUTPUT8 of OUTPUT-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the OUTPUT8 of OUTPUT-XBAR</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>



**9.3.4.25 OUTPUTLATCH Register (Offset = 30h) [reset = 0h]**

OUTPUTLATCH is shown in [Figure 9-55](#) and described in [Table 9-61](#).

Return to the [Summary Table](#).

Output X-BAR Output Latch

**Figure 9-55. OUTPUTLATCH Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 9-61. OUTPUTLATCH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R	0h	Records the OUTPUT8 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered  Refer to the Output X-BAR section of this chapter for more details.  Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
6	OUTPUT7	R	0h	Records the OUTPUT7 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered  Refer to the Output X-BAR section of this chapter for more details.  Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
5	OUTPUT6	R	0h	Records the OUTPUT6 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered  Refer to the Output X-BAR section of this chapter for more details.  Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn

**Table 9-61. OUTPUTLATCH Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	OUTPUT5	R	0h	<p>Records the OUTPUT5 of OUTPUT-XBAR.</p> <p>0: Respective output has not been triggered 1: Respective output is triggered</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
3	OUTPUT4	R	0h	<p>Records the OUTPUT4 of OUTPUT-XBAR.</p> <p>0: Respective output has not been triggered 1: Respective output is triggered</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
2	OUTPUT3	R	0h	<p>Records the OUTPUT3 of OUTPUT-XBAR.</p> <p>0: Respective output has not been triggered 1: Respective output is triggered</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
1	OUTPUT2	R	0h	<p>Records the OUTPUT2 of OUTPUT-XBAR.</p> <p>0: Respective output has not been triggered 1: Respective output is triggered</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>
0	OUTPUT1	R	0h	<p>Records the OUTPUT1 of OUTPUT-XBAR.</p> <p>0: Respective output has not been triggered 1: Respective output is triggered</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn</p>

### 9.3.4.26 OUTPUTLATCHCLR Register (Offset = 32h) [reset = 0h]

OUTPUTLATCHCLR is shown in [Figure 9-56](#) and described in [Table 9-62](#).

Return to the [Summary Table](#).

Output X-BAR Output Latch Clear

**Figure 9-56. OUTPUTLATCHCLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 9-62. OUTPUTLATCHCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R-0/W1S	0h	Clears the Output-Latch for OUTPUT8 of OUTPUT-XBAR  Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	OUTPUT7	R-0/W1S	0h	Clears the Output-Latch for OUTPUT7 of OUTPUT-XBAR  Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	OUTPUT6	R-0/W1S	0h	Clears the Output-Latch for OUTPUT6 of OUTPUT-XBAR  Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	OUTPUT5	R-0/W1S	0h	Clears the Output-Latch for OUTPUT5 of OUTPUT-XBAR  Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-62. OUTPUTLATCHCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	OUTPUT4	R-0/W1S	0h	<p>Clears the Output-Latch for OUTPUT4 of OUTPUT-XBAR</p> <p>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register</p> <p>Write of 0 has no effect</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
2	OUTPUT3	R-0/W1S	0h	<p>Clears the Output-Latch for OUTPUT3 of OUTPUT-XBAR</p> <p>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register</p> <p>Write of 0 has no effect</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
1	OUTPUT2	R-0/W1S	0h	<p>Clears the Output-Latch for OUTPUT2 of OUTPUT-XBAR</p> <p>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register</p> <p>Write of 0 has no effect</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
0	OUTPUT1	R-0/W1S	0h	<p>Clears the Output-Latch for OUTPUT1 of OUTPUT-XBAR</p> <p>Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register</p> <p>Write of 0 has no effect</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**9.3.4.27 OUTPUTLATCHFRC Register (Offset = 34h) [reset = 0h]**

OUTPUTLATCHFRC is shown in [Figure 9-57](#) and described in [Table 9-63](#).

Return to the [Summary Table](#).

Output X-BAR Output Latch Clear

**Figure 9-57. OUTPUTLATCHFRC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 9-63. OUTPUTLATCHFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R-0/W1S	0h	Sets the Output-Latch for OUTPUT8 of OUTPUT-XBAR  Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	OUTPUT7	R-0/W1S	0h	Sets the Output-Latch for OUTPUT7 of OUTPUT-XBAR  Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	OUTPUT6	R-0/W1S	0h	Sets the Output-Latch for OUTPUT6 of OUTPUT-XBAR  Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	OUTPUT5	R-0/W1S	0h	Sets the Output-Latch for OUTPUT5 of OUTPUT-XBAR  Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect  Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-63. OUTPUTLATCHFRC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	OUTPUT4	R-0/W1S	0h	<p>Sets the Output-Latch for OUTPUT4 of OUTPUT-XBAR</p> <p>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register</p> <p>Write of 0 has no effect</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
2	OUTPUT3	R-0/W1S	0h	<p>Sets the Output-Latch for OUTPUT3 of OUTPUT-XBAR</p> <p>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register</p> <p>Write of 0 has no effect</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
1	OUTPUT2	R-0/W1S	0h	<p>Sets the Output-Latch for OUTPUT2 of OUTPUT-XBAR</p> <p>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register</p> <p>Write of 0 has no effect</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
0	OUTPUT1	R-0/W1S	0h	<p>Sets the Output-Latch for OUTPUT1 of OUTPUT-XBAR</p> <p>Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register</p> <p>Write of 0 has no effect</p> <p>Refer to the Output X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

### 9.3.4.28 OUTPUTLATCHENABLE Register (Offset = 36h) [reset = 0h]

OUTPUTLATCHENABLE is shown in [Figure 9-58](#) and described in [Table 9-64](#).

Return to the [Summary Table](#).

Output X-BAR Output Latch Enable

**Figure 9-58. OUTPUTLATCHENABLE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
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 9-64. OUTPUTLATCHENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R/W	0h	Selects the output latch to drive OUTPUT8 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	OUTPUT7	R/W	0h	Selects the output latch to drive OUTPUT7 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	OUTPUT6	R/W	0h	Selects the output latch to drive OUTPUT6 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	OUTPUT5	R/W	0h	Selects the output latch to drive OUTPUT5 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-64. OUTPUTLATCHENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	OUTPUT4	R/W	0h	Selects the output latch to drive OUTPUT4 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	OUTPUT3	R/W	0h	Selects the output latch to drive OUTPUT3 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	OUTPUT2	R/W	0h	Selects the output latch to drive OUTPUT2 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	OUTPUT1	R/W	0h	Selects the output latch to drive OUTPUT1 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



### 9.3.4.29 OUTPUTINV Register (Offset = 38h) [reset = 0h]

OUTPUTINV is shown in [Figure 9-59](#) and described in [Table 9-65](#).

Return to the [Summary Table](#).

Output X-BAR Output Inversion

**Figure 9-59. OUTPUTINV Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
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 9-65. OUTPUTINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R/W	0h	Selects polarity for OUTPUT8 of OUTPUT-XBAR 0: drives active high output 1: drives active-low output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	OUTPUT7	R/W	0h	Selects polarity for OUTPUT7 of OUTPUT-XBAR 0: drives active high output 1: drives active-low output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	OUTPUT6	R/W	0h	Selects polarity for OUTPUT6 of OUTPUT-XBAR 0: drives active high output 1: drives active-low output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	OUTPUT5	R/W	0h	Selects polarity for OUTPUT5 of OUTPUT-XBAR 0: drives active high output 1: drives active-low output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-65. OUTPUTINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	OUTPUT4	R/W	0h	<p>Selects polarity for OUTPUT4 of OUTPUT-XBAR</p> <p>0: drives active high output 1: drives active-low output</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	OUTPUT3	R/W	0h	<p>Selects polarity for OUTPUT3 of OUTPUT-XBAR</p> <p>0: drives active high output 1: drives active-low output</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	OUTPUT2	R/W	0h	<p>Selects polarity for OUTPUT2 of OUTPUT-XBAR</p> <p>0: drives active high output 1: drives active-low output</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	OUTPUT1	R/W	0h	<p>Selects polarity for OUTPUT1 of OUTPUT-XBAR</p> <p>0: drives active high output 1: drives active-low output</p> <p>Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**9.3.4.30 OUTPUTLOCK Register (Offset = 3Eh) [reset = 0h]**

OUTPUTLOCK is shown in [Figure 9-60](#) and described in [Table 9-66](#).

Return to the [Summary Table](#).

Output X-BAR Configuration Lock register

**Figure 9-60. OUTPUTLOCK Register**

31	30	29	28	27	26	25	24
KEY							
R-0/W1S-0h							
23	22	21	20	19	18	17	16
KEY							
R-0/W1S-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED							LOCK
R-0-0h							R/WOnce-0h

**Table 9-66. OUTPUTLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W1S	0h	Bit-0 of this register can be set only if KEY= 0x5a5a Reset type: CPU1.SYSRSn
15-1	RESERVED	R-0	0h	Reserved
0	LOCK	R/WOnce	0h	Locks the configuration for OUTPUT-XBAR. Once the configuration is locked, writes to the below registers for OUTPUT-XBAR is blocked.  Registers Affected by the LOCK mechanism: OUTPUT-XBAROUTyMUX0TO15CFG OUTPUT-XBAROUTyMUX16TO31CFG OUTPUT-XBAROUTyMUXENABLE OUTPUT-XBAROUTLATENABLE OUTPUT-XBAROUTINV  0: Writes to the above registers are allowed 1: Writes to the above registers are blocked  Note: [1] LOCK mechanism only applies to writes. Reads are never blocked. Reset type: CPU1.SYSRSn

### 9.3.5 EPWM\_XBAR\_REGS Registers

Table 9-67 lists the EPWM\_XBAR\_REGS registers. All register offset addresses not listed in Table 9-67 should be considered as reserved locations and the register contents should not be modified.

**Table 9-67. EPWM\_XBAR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	TRIP4MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP4	EALLOW	<a href="#">Go</a>
2h	TRIP4MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP4	EALLOW	<a href="#">Go</a>
4h	TRIP5MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP5	EALLOW	<a href="#">Go</a>
6h	TRIP5MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP5	EALLOW	<a href="#">Go</a>
8h	TRIP7MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP7	EALLOW	<a href="#">Go</a>
Ah	TRIP7MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP7	EALLOW	<a href="#">Go</a>
Ch	TRIP8MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP8	EALLOW	<a href="#">Go</a>
Eh	TRIP8MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP8	EALLOW	<a href="#">Go</a>
10h	TRIP9MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP9	EALLOW	<a href="#">Go</a>
12h	TRIP9MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP9	EALLOW	<a href="#">Go</a>
14h	TRIP10MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP10	EALLOW	<a href="#">Go</a>
16h	TRIP10MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP10	EALLOW	<a href="#">Go</a>
18h	TRIP11MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP11	EALLOW	<a href="#">Go</a>
1Ah	TRIP11MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP11	EALLOW	<a href="#">Go</a>
1Ch	TRIP12MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP12	EALLOW	<a href="#">Go</a>
1Eh	TRIP12MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP12	EALLOW	<a href="#">Go</a>
20h	TRIP4MUXENABLE	ePWM XBAR Mux Enable for TRIP4	EALLOW	<a href="#">Go</a>
22h	TRIP5MUXENABLE	ePWM XBAR Mux Enable for TRIP5	EALLOW	<a href="#">Go</a>
24h	TRIP7MUXENABLE	ePWM XBAR Mux Enable for TRIP7	EALLOW	<a href="#">Go</a>
26h	TRIP8MUXENABLE	ePWM XBAR Mux Enable for TRIP8	EALLOW	<a href="#">Go</a>
28h	TRIP9MUXENABLE	ePWM XBAR Mux Enable for TRIP9	EALLOW	<a href="#">Go</a>
2Ah	TRIP10MUXENABLE	ePWM XBAR Mux Enable for TRIP10	EALLOW	<a href="#">Go</a>
2Ch	TRIP11MUXENABLE	ePWM XBAR Mux Enable for TRIP11	EALLOW	<a href="#">Go</a>
2Eh	TRIP12MUXENABLE	ePWM XBAR Mux Enable for TRIP12	EALLOW	<a href="#">Go</a>
38h	TRIPOUTINV	ePWM XBAR Output Inversion Register	EALLOW	<a href="#">Go</a>
3Eh	TRIPLOCK	ePWM XBAR Configuration Lock register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 9-68 shows the codes that are used for access types in this section.

**Table 9-68. EPWM\_XBAR\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
WOnce	WOnce	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		

**Table 9-68. EPWM\_XBAR\_REGS Access Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**9.3.5.1 TRIP4MUX0TO15CFG Register (Offset = 0h) [reset = 0h]**

TRIP4MUX0TO15CFG is shown in [Figure 9-61](#) and described in [Table 9-69](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP4

**Figure 9-61. TRIP4MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-69. TRIP4MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-69. TRIP4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-69. TRIP4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



### 9.3.5.2 TRIP4MUX16TO31CFG Register (Offset = 2h) [reset = 0h]

TRIP4MUX16TO31CFG is shown in [Figure 9-62](#) and described in [Table 9-70](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP4

**Figure 9-62. TRIP4MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-70. TRIP4MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-70. TRIP4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux27:</p> <p>00 : Select .0 input for Mux27  01 : Select .1 input for Mux27  10 : Select .2 input for Mux27  11 : Select .3 input for Mux27</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux26:</p> <p>00 : Select .0 input for Mux26  01 : Select .1 input for Mux26  10 : Select .2 input for Mux26  11 : Select .3 input for Mux26</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux25:</p> <p>00 : Select .0 input for Mux25  01 : Select .1 input for Mux25  10 : Select .2 input for Mux25  11 : Select .3 input for Mux25</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux24:</p> <p>00 : Select .0 input for Mux24  01 : Select .1 input for Mux24  10 : Select .2 input for Mux24  11 : Select .3 input for Mux24</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux23:</p> <p>00 : Select .0 input for Mux23  01 : Select .1 input for Mux23  10 : Select .2 input for Mux23  11 : Select .3 input for Mux23</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
13-12	MUX22	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux22:</p> <p>00 : Select .0 input for Mux22  01 : Select .1 input for Mux22  10 : Select .2 input for Mux22  11 : Select .3 input for Mux22</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-70. TRIP4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux21:</p> <p>00 : Select .0 input for Mux21            01 : Select .1 input for Mux21            10 : Select .2 input for Mux21            11 : Select .3 input for Mux21</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux20:</p> <p>00 : Select .0 input for Mux20            01 : Select .1 input for Mux20            10 : Select .2 input for Mux20            11 : Select .3 input for Mux20</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux19:</p> <p>00 : Select .0 input for Mux19            01 : Select .1 input for Mux19            10 : Select .2 input for Mux19            11 : Select .3 input for Mux19</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux18:</p> <p>00 : Select .0 input for Mux18            01 : Select .1 input for Mux18            10 : Select .2 input for Mux18            11 : Select .3 input for Mux18</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
3-2	MUX17	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux17:</p> <p>00 : Select .0 input for Mux17            01 : Select .1 input for Mux17            10 : Select .2 input for Mux17            11 : Select .3 input for Mux17</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP4 Mux16:</p> <p>00 : Select .0 input for Mux16            01 : Select .1 input for Mux16            10 : Select .2 input for Mux16            11 : Select .3 input for Mux16</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

### 9.3.5.3 TRIP5MUX0TO15CFG Register (Offset = 4h) [reset = 0h]

TRIP5MUX0TO15CFG is shown in [Figure 9-63](#) and described in [Table 9-71](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP5

**Figure 9-63. TRIP5MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-71. TRIP5MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-71. TRIP5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP5 Mux11:</p> <p>00 : Select .0 input for Mux11            01 : Select .1 input for Mux11            10 : Select .2 input for Mux11            11 : Select .3 input for Mux11</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP5 Mux10:</p> <p>00 : Select .0 input for Mux10            01 : Select .1 input for Mux10            10 : Select .2 input for Mux10            11 : Select .3 input for Mux10</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP5 Mux9:</p> <p>00 : Select .0 input for Mux9            01 : Select .1 input for Mux9            10 : Select .2 input for Mux9            11 : Select .3 input for Mux9</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP5 Mux8:</p> <p>00 : Select .0 input for Mux8            01 : Select .1 input for Mux8            10 : Select .2 input for Mux8            11 : Select .3 input for Mux8</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP5 Mux7:</p> <p>00 : Select .0 input for Mux7            01 : Select .1 input for Mux7            10 : Select .2 input for Mux7            11 : Select .3 input for Mux7</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
13-12	MUX6	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP5 Mux6:</p> <p>00 : Select .0 input for Mux6            01 : Select .1 input for Mux6            10 : Select .2 input for Mux6            11 : Select .3 input for Mux6</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-71. TRIP5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.4 TRIP5MUX16TO31CFG Register (Offset = 6h) [reset = 0h]

TRIP5MUX16TO31CFG is shown in [Figure 9-64](#) and described in [Table 9-72](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP5

**Figure 9-64. TRIP5MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-72. TRIP5MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-72. TRIP5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



**Table 9-72. TRIP5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.5 TRIP7MUX0TO15CFG Register (Offset = 8h) [reset = 0h]

TRIP7MUX0TO15CFG is shown in [Figure 9-65](#) and described in [Table 9-73](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP7

**Figure 9-65. TRIP7MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-73. TRIP7MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-73. TRIP7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux11:</p> <p>00 : Select .0 input for Mux11            01 : Select .1 input for Mux11            10 : Select .2 input for Mux11            11 : Select .3 input for Mux11</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux10:</p> <p>00 : Select .0 input for Mux10            01 : Select .1 input for Mux10            10 : Select .2 input for Mux10            11 : Select .3 input for Mux10</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux9:</p> <p>00 : Select .0 input for Mux9            01 : Select .1 input for Mux9            10 : Select .2 input for Mux9            11 : Select .3 input for Mux9</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux8:</p> <p>00 : Select .0 input for Mux8            01 : Select .1 input for Mux8            10 : Select .2 input for Mux8            11 : Select .3 input for Mux8</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux7:</p> <p>00 : Select .0 input for Mux7            01 : Select .1 input for Mux7            10 : Select .2 input for Mux7            11 : Select .3 input for Mux7</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
13-12	MUX6	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux6:</p> <p>00 : Select .0 input for Mux6            01 : Select .1 input for Mux6            10 : Select .2 input for Mux6            11 : Select .3 input for Mux6</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-73. TRIP7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux5:</p> <p>00 : Select .0 input for Mux5  01 : Select .1 input for Mux5  10 : Select .2 input for Mux5  11 : Select .3 input for Mux5</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux4:</p> <p>00 : Select .0 input for Mux4  01 : Select .1 input for Mux4  10 : Select .2 input for Mux4  11 : Select .3 input for Mux4</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux3:</p> <p>00 : Select .0 input for Mux3  01 : Select .1 input for Mux3  10 : Select .2 input for Mux3  11 : Select .3 input for Mux3</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux2:</p> <p>00 : Select .0 input for Mux2  01 : Select .1 input for Mux2  10 : Select .2 input for Mux2  11 : Select .3 input for Mux2</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
3-2	MUX1	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux1:</p> <p>00 : Select .0 input for Mux1  01 : Select .1 input for Mux1  10 : Select .2 input for Mux1  11 : Select .3 input for Mux1</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP7 Mux0:</p> <p>00 : Select .0 input for Mux0  01 : Select .1 input for Mux0  10 : Select .2 input for Mux0  11 : Select .3 input for Mux0</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.5.6 TRIP7MUX16TO31CFG Register (Offset = Ah) [reset = 0h]

TRIP7MUX16TO31CFG is shown in [Figure 9-66](#) and described in [Table 9-74](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP7

**Figure 9-66. TRIP7MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-74. TRIP7MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-74. TRIP7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-74. TRIP7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.7 TRIP8MUX0TO15CFG Register (Offset = Ch) [reset = 0h]

TRIP8MUX0TO15CFG is shown in [Figure 9-67](#) and described in [Table 9-75](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP8

**Figure 9-67. TRIP8MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-75. TRIP8MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



**Table 9-75. TRIP8MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-75. TRIP8MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux5:</p> <p>00 : Select .0 input for Mux5  01 : Select .1 input for Mux5  10 : Select .2 input for Mux5  11 : Select .3 input for Mux5</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux4:</p> <p>00 : Select .0 input for Mux4  01 : Select .1 input for Mux4  10 : Select .2 input for Mux4  11 : Select .3 input for Mux4</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux3:</p> <p>00 : Select .0 input for Mux3  01 : Select .1 input for Mux3  10 : Select .2 input for Mux3  11 : Select .3 input for Mux3</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux2:</p> <p>00 : Select .0 input for Mux2  01 : Select .1 input for Mux2  10 : Select .2 input for Mux2  11 : Select .3 input for Mux2</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
3-2	MUX1	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux1:</p> <p>00 : Select .0 input for Mux1  01 : Select .1 input for Mux1  10 : Select .2 input for Mux1  11 : Select .3 input for Mux1</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux0:</p> <p>00 : Select .0 input for Mux0  01 : Select .1 input for Mux0  10 : Select .2 input for Mux0  11 : Select .3 input for Mux0</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.5.8 TRIP8MUX16TO31CFG Register (Offset = Eh) [reset = 0h]

TRIP8MUX16TO31CFG is shown in [Figure 9-68](#) and described in [Table 9-76](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP8

**Figure 9-68. TRIP8MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-76. TRIP8MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-76. TRIP8MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux27:</p> <p>00 : Select .0 input for Mux27  01 : Select .1 input for Mux27  10 : Select .2 input for Mux27  11 : Select .3 input for Mux27</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux26:</p> <p>00 : Select .0 input for Mux26  01 : Select .1 input for Mux26  10 : Select .2 input for Mux26  11 : Select .3 input for Mux26</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux25:</p> <p>00 : Select .0 input for Mux25  01 : Select .1 input for Mux25  10 : Select .2 input for Mux25  11 : Select .3 input for Mux25</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux24:</p> <p>00 : Select .0 input for Mux24  01 : Select .1 input for Mux24  10 : Select .2 input for Mux24  11 : Select .3 input for Mux24</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux23:</p> <p>00 : Select .0 input for Mux23  01 : Select .1 input for Mux23  10 : Select .2 input for Mux23  11 : Select .3 input for Mux23</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
13-12	MUX22	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux22:</p> <p>00 : Select .0 input for Mux22  01 : Select .1 input for Mux22  10 : Select .2 input for Mux22  11 : Select .3 input for Mux22</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-76. TRIP8MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux21:</p> <p>00 : Select .0 input for Mux21            01 : Select .1 input for Mux21            10 : Select .2 input for Mux21            11 : Select .3 input for Mux21</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux20:</p> <p>00 : Select .0 input for Mux20            01 : Select .1 input for Mux20            10 : Select .2 input for Mux20            11 : Select .3 input for Mux20</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux19:</p> <p>00 : Select .0 input for Mux19            01 : Select .1 input for Mux19            10 : Select .2 input for Mux19            11 : Select .3 input for Mux19</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux18:</p> <p>00 : Select .0 input for Mux18            01 : Select .1 input for Mux18            10 : Select .2 input for Mux18            11 : Select .3 input for Mux18</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
3-2	MUX17	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux17:</p> <p>00 : Select .0 input for Mux17            01 : Select .1 input for Mux17            10 : Select .2 input for Mux17            11 : Select .3 input for Mux17</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP8 Mux16:</p> <p>00 : Select .0 input for Mux16            01 : Select .1 input for Mux16            10 : Select .2 input for Mux16            11 : Select .3 input for Mux16</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

### 9.3.5.9 TRIP9MUX0TO15CFG Register (Offset = 10h) [reset = 0h]

TRIP9MUX0TO15CFG is shown in [Figure 9-69](#) and described in [Table 9-77](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP9

**Figure 9-69. TRIP9MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-77. TRIP9MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-77. TRIP9MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux11:</p> <p>00 : Select .0 input for Mux11            01 : Select .1 input for Mux11            10 : Select .2 input for Mux11            11 : Select .3 input for Mux11</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux10:</p> <p>00 : Select .0 input for Mux10            01 : Select .1 input for Mux10            10 : Select .2 input for Mux10            11 : Select .3 input for Mux10</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux9:</p> <p>00 : Select .0 input for Mux9            01 : Select .1 input for Mux9            10 : Select .2 input for Mux9            11 : Select .3 input for Mux9</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux8:</p> <p>00 : Select .0 input for Mux8            01 : Select .1 input for Mux8            10 : Select .2 input for Mux8            11 : Select .3 input for Mux8</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux7:</p> <p>00 : Select .0 input for Mux7            01 : Select .1 input for Mux7            10 : Select .2 input for Mux7            11 : Select .3 input for Mux7</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
13-12	MUX6	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux6:</p> <p>00 : Select .0 input for Mux6            01 : Select .1 input for Mux6            10 : Select .2 input for Mux6            11 : Select .3 input for Mux6</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-77. TRIP9MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux5:</p> <p>00 : Select .0 input for Mux5  01 : Select .1 input for Mux5  10 : Select .2 input for Mux5  11 : Select .3 input for Mux5</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux4:</p> <p>00 : Select .0 input for Mux4  01 : Select .1 input for Mux4  10 : Select .2 input for Mux4  11 : Select .3 input for Mux4</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux3:</p> <p>00 : Select .0 input for Mux3  01 : Select .1 input for Mux3  10 : Select .2 input for Mux3  11 : Select .3 input for Mux3</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux2:</p> <p>00 : Select .0 input for Mux2  01 : Select .1 input for Mux2  10 : Select .2 input for Mux2  11 : Select .3 input for Mux2</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
3-2	MUX1	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux1:</p> <p>00 : Select .0 input for Mux1  01 : Select .1 input for Mux1  10 : Select .2 input for Mux1  11 : Select .3 input for Mux1</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP9 Mux0:</p> <p>00 : Select .0 input for Mux0  01 : Select .1 input for Mux0  10 : Select .2 input for Mux0  11 : Select .3 input for Mux0</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



### 9.3.5.10 TRIP9MUX16TO31CFG Register (Offset = 12h) [reset = 0h]

TRIP9MUX16TO31CFG is shown in [Figure 9-70](#) and described in [Table 9-78](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP9

**Figure 9-70. TRIP9MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-78. TRIP9MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-78. TRIP9MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-78. TRIP9MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**9.3.5.11 TRIP10MUX0TO15CFG Register (Offset = 14h) [reset = 0h]**

TRIP10MUX0TO15CFG is shown in [Figure 9-71](#) and described in [Table 9-79](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP10

**Figure 9-71. TRIP10MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-79. TRIP10MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-79. TRIP10MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-79. TRIP10MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux5:</p> <p>00 : Select .0 input for Mux5  01 : Select .1 input for Mux5  10 : Select .2 input for Mux5  11 : Select .3 input for Mux5</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux4:</p> <p>00 : Select .0 input for Mux4  01 : Select .1 input for Mux4  10 : Select .2 input for Mux4  11 : Select .3 input for Mux4</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux3:</p> <p>00 : Select .0 input for Mux3  01 : Select .1 input for Mux3  10 : Select .2 input for Mux3  11 : Select .3 input for Mux3</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux2:</p> <p>00 : Select .0 input for Mux2  01 : Select .1 input for Mux2  10 : Select .2 input for Mux2  11 : Select .3 input for Mux2</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
3-2	MUX1	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux1:</p> <p>00 : Select .0 input for Mux1  01 : Select .1 input for Mux1  10 : Select .2 input for Mux1  11 : Select .3 input for Mux1</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux0:</p> <p>00 : Select .0 input for Mux0  01 : Select .1 input for Mux0  10 : Select .2 input for Mux0  11 : Select .3 input for Mux0</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.5.12 TRIP10MUX16TO31CFG Register (Offset = 16h) [reset = 0h]

TRIP10MUX16TO31CFG is shown in [Figure 9-72](#) and described in [Table 9-80](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP10

**Figure 9-72. TRIP10MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-80. TRIP10MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-80. TRIP10MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux27:</p> <p>00 : Select .0 input for Mux27  01 : Select .1 input for Mux27  10 : Select .2 input for Mux27  11 : Select .3 input for Mux27</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux26:</p> <p>00 : Select .0 input for Mux26  01 : Select .1 input for Mux26  10 : Select .2 input for Mux26  11 : Select .3 input for Mux26</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux25:</p> <p>00 : Select .0 input for Mux25  01 : Select .1 input for Mux25  10 : Select .2 input for Mux25  11 : Select .3 input for Mux25</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux24:</p> <p>00 : Select .0 input for Mux24  01 : Select .1 input for Mux24  10 : Select .2 input for Mux24  11 : Select .3 input for Mux24</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux23:</p> <p>00 : Select .0 input for Mux23  01 : Select .1 input for Mux23  10 : Select .2 input for Mux23  11 : Select .3 input for Mux23</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
13-12	MUX22	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP10 Mux22:</p> <p>00 : Select .0 input for Mux22  01 : Select .1 input for Mux22  10 : Select .2 input for Mux22  11 : Select .3 input for Mux22</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



**Table 9-80. TRIP10MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux21:  00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux20:  00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux19:  00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux18:  00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux17:  00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux16:  00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.13 TRIP11MUX0TO15CFG Register (Offset = 18h) [reset = 0h]

TRIP11MUX0TO15CFG is shown in [Figure 9-73](#) and described in [Table 9-81](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP11

**Figure 9-73. TRIP11MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-81. TRIP11MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-81. TRIP11MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux11:</p> <p>00 : Select .0 input for Mux11            01 : Select .1 input for Mux11            10 : Select .2 input for Mux11            11 : Select .3 input for Mux11</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux10:</p> <p>00 : Select .0 input for Mux10            01 : Select .1 input for Mux10            10 : Select .2 input for Mux10            11 : Select .3 input for Mux10</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux9:</p> <p>00 : Select .0 input for Mux9            01 : Select .1 input for Mux9            10 : Select .2 input for Mux9            11 : Select .3 input for Mux9</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux8:</p> <p>00 : Select .0 input for Mux8            01 : Select .1 input for Mux8            10 : Select .2 input for Mux8            11 : Select .3 input for Mux8</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux7:</p> <p>00 : Select .0 input for Mux7            01 : Select .1 input for Mux7            10 : Select .2 input for Mux7            11 : Select .3 input for Mux7</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
13-12	MUX6	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux6:</p> <p>00 : Select .0 input for Mux6            01 : Select .1 input for Mux6            10 : Select .2 input for Mux6            11 : Select .3 input for Mux6</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-81. TRIP11MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.14 TRIP11MUX16TO31CFG Register (Offset = 1Ah) [reset = 0h]

TRIP11MUX16TO31CFG is shown in [Figure 9-74](#) and described in [Table 9-82](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP11

**Figure 9-74. TRIP11MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-82. TRIP11MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-82. TRIP11MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux27:</p> <p>00 : Select .0 input for Mux27  01 : Select .1 input for Mux27  10 : Select .2 input for Mux27  11 : Select .3 input for Mux27</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux26:</p> <p>00 : Select .0 input for Mux26  01 : Select .1 input for Mux26  10 : Select .2 input for Mux26  11 : Select .3 input for Mux26</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux25:</p> <p>00 : Select .0 input for Mux25  01 : Select .1 input for Mux25  10 : Select .2 input for Mux25  11 : Select .3 input for Mux25</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux24:</p> <p>00 : Select .0 input for Mux24  01 : Select .1 input for Mux24  10 : Select .2 input for Mux24  11 : Select .3 input for Mux24</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux23:</p> <p>00 : Select .0 input for Mux23  01 : Select .1 input for Mux23  10 : Select .2 input for Mux23  11 : Select .3 input for Mux23</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
13-12	MUX22	R/W	0h	<p>Select Bits for EPWM-XBAR TRIP11 Mux22:</p> <p>00 : Select .0 input for Mux22  01 : Select .1 input for Mux22  10 : Select .2 input for Mux22  11 : Select .3 input for Mux22</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-82. TRIP11MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux21:  00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux20:  00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux19:  00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux18:  00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux17:  00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux16:  00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.15 TRIP12MUX0TO15CFG Register (Offset = 1Ch) [reset = 0h]

TRIP12MUX0TO15CFG is shown in [Figure 9-75](#) and described in [Table 9-83](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP12

**Figure 9-75. TRIP12MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-83. TRIP12MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux15:  00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux14:  00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux13:  00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux12:  00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn



**Table 9-83. TRIP12MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-83. TRIP12MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.16 TRIP12MUX16TO31CFG Register (Offset = 1Eh) [reset = 0h]

TRIP12MUX16TO31CFG is shown in [Figure 9-76](#) and described in [Table 9-84](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP12

**Figure 9-76. TRIP12MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-84. TRIP12MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux31:  00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux30:  00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux29:  00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux28:  00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-84. TRIP12MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-84. TRIP12MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux21:  00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux20:  00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux19:  00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux18:  00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux17:  00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux16:  00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.17 TRIP4MUXENABLE Register (Offset = 20h) [reset = 0h]

TRIP4MUXENABLE is shown in [Figure 9-77](#) and described in [Table 9-85](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP4

**Figure 9-77. TRIP4MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
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
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
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
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
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 9-85. TRIP4MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP4 of EPWM-XBAR  0: Respective output of Mux31 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP4 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP4 of EPWM-XBAR  0: Respective output of Mux30 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP4 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP4 of EPWM-XBAR  0: Respective output of Mux29 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP4 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP4 of EPWM-XBAR  0: Respective output of Mux28 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP4 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-85. TRIP4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-85. TRIP4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>



**Table 9-85. TRIP4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-85. TRIP4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of Mux0 to drive TRIP4 of EPWM-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the TRIP4 of EPWM-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the TRIP4 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.5.18 TRIP5MUXENABLE Register (Offset = 22h) [reset = 0h]

TRIP5MUXENABLE is shown in [Figure 9-78](#) and described in [Table 9-86](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP5

**Figure 9-78. TRIP5MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-86. TRIP5MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of Mux31 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux31 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux31 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of Mux30 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux30 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux30 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of Mux29 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux29 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux29 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of Mux28 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux28 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux28 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-86. TRIP5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-86. TRIP5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-86. TRIP5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-86. TRIP5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive TRIP5 of EPWM-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the TRIP5 of EPWM-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the TRIP5 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.5.19 TRIP7MUXENABLE Register (Offset = 24h) [reset = 0h]

TRIP7MUXENABLE is shown in [Figure 9-79](#) and described in [Table 9-87](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP7

**Figure 9-79. TRIP7MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-87. TRIP7MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of Mux31 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux31 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux31 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of Mux30 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux30 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux30 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of Mux29 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux29 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux29 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of Mux28 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux28 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux28 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>



**Table 9-87. TRIP7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-87. TRIP7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-87. TRIP7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-87. TRIP7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive TRIP7 of EPWM-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the TRIP7 of EPWM-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the TRIP7 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.5.20 TRIP8MUXENABLE Register (Offset = 26h) [reset = 0h]

TRIP8MUXENABLE is shown in [Figure 9-80](#) and described in [Table 9-88](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP8

**Figure 9-80. TRIP8MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-88. TRIP8MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of Mux31 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux31 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux31 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of Mux30 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux30 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux30 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of Mux29 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux29 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux29 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of Mux28 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux28 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux28 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-88. TRIP8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-88. TRIP8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-88. TRIP8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>



**Table 9-88. TRIP8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive TRIP8 of EPWM-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the TRIP8 of EPWM-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the TRIP8 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.5.21 TRIP9MUXENABLE Register (Offset = 28h) [reset = 0h]

TRIP9MUXENABLE is shown in [Figure 9-81](#) and described in [Table 9-89](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP9

**Figure 9-81. TRIP9MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
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
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
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
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
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 9-89. TRIP9MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP9 of EPWM-XBAR  0: Respective output of Mux31 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP9 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP9 of EPWM-XBAR  0: Respective output of Mux30 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP9 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP9 of EPWM-XBAR  0: Respective output of Mux29 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP9 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP9 of EPWM-XBAR  0: Respective output of Mux28 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP9 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-89. TRIP9MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-89. TRIP9MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-89. TRIP9MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-89. TRIP9MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive TRIP9 of EPWM-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the TRIP9 of EPWM-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the TRIP9 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.5.22 TRIP10MUXENABLE Register (Offset = 2Ah) [reset = 0h]

TRIP10MUXENABLE is shown in [Figure 9-82](#) and described in [Table 9-90](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP10

**Figure 9-82. TRIP10MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
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
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
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
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
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 9-90. TRIP10MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP10 of EPWM-XBAR  0: Respective output of Mux31 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP10 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP10 of EPWM-XBAR  0: Respective output of Mux30 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP10 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP10 of EPWM-XBAR  0: Respective output of Mux29 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP10 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP10 of EPWM-XBAR  0: Respective output of Mux28 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP10 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-90. TRIP10MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>



**Table 9-90. TRIP10MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-90. TRIP10MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-90. TRIP10MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive TRIP10 of EPWM-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the TRIP10 of EPWM-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the TRIP10 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**9.3.5.23 TRIP11MUXENABLE Register (Offset = 2Ch) [reset = 0h]**

TRIP11MUXENABLE is shown in [Figure 9-83](#) and described in [Table 9-91](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP11

**Figure 9-83. TRIP11MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-91. TRIP11MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP11 of EPWM-XBAR  0: Respective output of Mux31 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP11 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP11 of EPWM-XBAR  0: Respective output of Mux30 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP11 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP11 of EPWM-XBAR  0: Respective output of Mux29 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP11 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP11 of EPWM-XBAR  0: Respective output of Mux28 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP11 of EPWM-XBAR  Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-91. TRIP11MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-91. TRIP11MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-91. TRIP11MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-91. TRIP11MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive TRIP11 of EPWM-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the TRIP11 of EPWM-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the TRIP11 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>



### 9.3.5.24 TRIP12MUXENABLE Register (Offset = 2Eh) [reset = 0h]

TRIP12MUXENABLE is shown in [Figure 9-84](#) and described in [Table 9-92](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP12

**Figure 9-84. TRIP12MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
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
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
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
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
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 9-92. TRIP12MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of Mux31 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux31 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux31 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of Mux30 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux30 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux30 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of Mux29 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux29 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux29 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of Mux28 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux28 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux28 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-92. TRIP12MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of Mux27 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux27 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux27 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of Mux26 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux26 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux26 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of Mux25 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux25 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux25 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of Mux24 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux24 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux24 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of Mux23 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux23 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux23 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of Mux22 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux22 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux22 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of Mux21 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux21 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux21 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-92. TRIP12MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of Mux20 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux20 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux20 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of Mux19 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux19 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux19 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of Mux18 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux18 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux18 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of Mux17 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux17 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux17 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of Mux16 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux16 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux16 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of Mux15 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux15 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux15 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of Mux14 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux14 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux14 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-92. TRIP12MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of Mux13 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux13 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux13 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of Mux12 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux12 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux12 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of Mux11 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux11 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux11 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of Mux10 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux10 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux10 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of Mux9 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux9 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux9 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of Mux8 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux8 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux8 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of Mux7 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux7 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux7 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-92. TRIP12MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of Mux6 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux6 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux6 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of Mux5 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux5 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux5 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of Mux4 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux4 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux4 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of Mux3 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux3 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux3 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of Mux2 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux2 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux2 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of Mux1 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux1 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux1 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive TRIP12 of EPWM-XBAR</p> <p>0: Respective output of Mux0 is disabled to drive the TRIP12 of EPWM-XBAR</p> <p>1: Respective output of Mux0 is enabled to drive the TRIP12 of EPWM-XBAR</p> <p>Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**9.3.5.25 TRIPOUTINV Register (Offset = 38h) [reset = 0h]**

TRIP0UTINV is shown in [Figure 9-85](#) and described in [Table 9-93](#).

Return to the [Summary Table](#).

ePWM XBAR Output Inversion Register

**Figure 9-85. TRIPOUTINV Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
TRIP12	TRIP11	TRIP10	TRIP9	TRIP8	TRIP7	TRIP5	TRIP4
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 9-93. TRIPOUTINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	TRIP12	R/W	0h	Selects polarity for TRIP12 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	TRIP11	R/W	0h	Selects polarity for TRIP11 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	TRIP10	R/W	0h	Selects polarity for TRIP10 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	TRIP9	R/W	0h	Selects polarity for TRIP9 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-93. TRIPOUTINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	TRIP8	R/W	0h	Selects polarity for TRIP8 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	TRIP7	R/W	0h	Selects polarity for TRIP7 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	TRIP5	R/W	0h	Selects polarity for TRIP5 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	TRIP4	R/W	0h	Selects polarity for TRIP4 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**9.3.5.26 TRIPLOCK Register (Offset = 3Eh) [reset = 0h]**

TRIPLOCK is shown in [Figure 9-86](#) and described in [Table 9-94](#).

Return to the [Summary Table](#).

ePWM XBAR Configuration Lock register

**Figure 9-86. TRIPLOCK Register**

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

**Table 9-94. TRIPLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	Bit-0 of this register can be set only if KEY= 0x5a5a Reset type: CPU1.SYSRSn
15-1	RESERVED	R-0	0h	Reserved
0	LOCK	R/WOnce	0h	Locks the configuration for EPWM-XBAR. Once the configuration is locked, writes to the below registers for EPWM-XBAR is blocked.  Registers Affected by the LOCK mechanism: EPWM-XBAROUTyMUX0TO15CFG EPWM-XBAROUTyMUX16TO31CFG EPWM-XBAROUTyMUXENABLE EPWM-XBAROUTLATEN EPWM-XBAROUTINV  0: Writes to the above registers are allowed 1: Writes to the above registers are blocked  Note: [1] LOCK mechanism only applies to writes. Reads are never blocked.  Reset type: CPU1.SYSRSn



### 9.3.6 CLB\_XBAR\_REGS Registers

Table 9-95 lists the CLB\_XBAR\_REGS registers. All register offset addresses not listed in Table 9-95 should be considered as reserved locations and the register contents should not be modified.

**Table 9-95. CLB\_XBAR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	AUXSIG0MUX0TO15CFG	CLB XBAR Mux Configuration for Output-0	EALLOW	<a href="#">Go</a>
2h	AUXSIG0MUX16TO31CFG	CLB XBAR Mux Configuration for Output-0	EALLOW	<a href="#">Go</a>
4h	AUXSIG1MUX0TO15CFG	CLB XBAR Mux Configuration for Output-1	EALLOW	<a href="#">Go</a>
6h	AUXSIG1MUX16TO31CFG	CLB XBAR Mux Configuration for Output-1	EALLOW	<a href="#">Go</a>
8h	AUXSIG2MUX0TO15CFG	CLB XBAR Mux Configuration for Output-2	EALLOW	<a href="#">Go</a>
Ah	AUXSIG2MUX16TO31CFG	CLB XBAR Mux Configuration for Output-2	EALLOW	<a href="#">Go</a>
Ch	AUXSIG3MUX0TO15CFG	CLB XBAR Mux Configuration for Output-3	EALLOW	<a href="#">Go</a>
Eh	AUXSIG3MUX16TO31CFG	CLB XBAR Mux Configuration for Output-3	EALLOW	<a href="#">Go</a>
10h	AUXSIG4MUX0TO15CFG	CLB XBAR Mux Configuration for Output-4	EALLOW	<a href="#">Go</a>
12h	AUXSIG4MUX16TO31CFG	CLB XBAR Mux Configuration for Output-4	EALLOW	<a href="#">Go</a>
14h	AUXSIG5MUX0TO15CFG	CLB XBAR Mux Configuration for Output-5	EALLOW	<a href="#">Go</a>
16h	AUXSIG5MUX16TO31CFG	CLB XBAR Mux Configuration for Output-5	EALLOW	<a href="#">Go</a>
18h	AUXSIG6MUX0TO15CFG	CLB XBAR Mux Configuration for Output-6	EALLOW	<a href="#">Go</a>
1Ah	AUXSIG6MUX16TO31CFG	CLB XBAR Mux Configuration for Output-6	EALLOW	<a href="#">Go</a>
1Ch	AUXSIG7MUX0TO15CFG	CLB XBAR Mux Configuration for Output-7	EALLOW	<a href="#">Go</a>
1Eh	AUXSIG7MUX16TO31CFG	CLB XBAR Mux Configuration for Output-7	EALLOW	<a href="#">Go</a>
20h	AUXSIG0MUXENABLE	CLB XBAR Mux Enable Register for Output-0	EALLOW	<a href="#">Go</a>
22h	AUXSIG1MUXENABLE	CLB XBAR Mux Enable Register for Output-1	EALLOW	<a href="#">Go</a>
24h	AUXSIG2MUXENABLE	CLB XBAR Mux Enable Register for Output-2	EALLOW	<a href="#">Go</a>
26h	AUXSIG3MUXENABLE	CLB XBAR Mux Enable Register for Output-3	EALLOW	<a href="#">Go</a>
28h	AUXSIG4MUXENABLE	CLB XBAR Mux Enable Register for Output-4	EALLOW	<a href="#">Go</a>
2Ah	AUXSIG5MUXENABLE	CLB XBAR Mux Enable Register for Output-5	EALLOW	<a href="#">Go</a>
2Ch	AUXSIG6MUXENABLE	CLB XBAR Mux Enable Register for Output-6	EALLOW	<a href="#">Go</a>
2Eh	AUXSIG7MUXENABLE	CLB XBAR Mux Enable Register for Output-7	EALLOW	<a href="#">Go</a>
38h	AUXSIGOUTINV	CLB XBAR Output Inversion Register	EALLOW	<a href="#">Go</a>
3Eh	AUXSIGLOCK	ClbXbar Configuration Lock register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 9-96 shows the codes that are used for access types in this section.

**Table 9-96. CLB\_XBAR\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
WOnce	W Once	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		

**Table 9-96. CLB\_XBAR\_REGS Access Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 9.3.6.1 AUXSIG0MUX0TO15CFG Register (Offset = 0h) [reset = 0h]

AUXSIG0MUX0TO15CFG is shown in [Figure 9-87](#) and described in [Table 9-97](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-0

**Figure 9-87. AUXSIG0MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-97. AUXSIG0MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG0 of CLB-XBAR  00 : Select .0 input for MUX15 01 : Select .1 input for MUX15 10 : Select .2 input for MUX15 11 : Select .3 input for MUX15  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG0 of CLB-XBAR  00 : Select .0 input for MUX14 01 : Select .1 input for MUX14 10 : Select .2 input for MUX14 11 : Select .3 input for MUX14  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG0 of CLB-XBAR  00 : Select .0 input for MUX13 01 : Select .1 input for MUX13 10 : Select .2 input for MUX13 11 : Select .3 input for MUX13  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG0 of CLB-XBAR  00 : Select .0 input for MUX12 01 : Select .1 input for MUX12 10 : Select .2 input for MUX12 11 : Select .3 input for MUX12  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-97. AUXSIG0MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX11  01 : Select .1 input for MUX11  10 : Select .2 input for MUX11  11 : Select .3 input for MUX11</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX10  01 : Select .1 input for MUX10  10 : Select .2 input for MUX10  11 : Select .3 input for MUX10</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX9  01 : Select .1 input for MUX9  10 : Select .2 input for MUX9  11 : Select .3 input for MUX9</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX8  01 : Select .1 input for MUX8  10 : Select .2 input for MUX8  11 : Select .3 input for MUX8</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX7  01 : Select .1 input for MUX7  10 : Select .2 input for MUX7  11 : Select .3 input for MUX7</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-97. AUXSIG0MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX6	R/W	0h	<p>Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX6            01 : Select .1 input for MUX6            10 : Select .2 input for MUX6            11 : Select .3 input for MUX6</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX5	R/W	0h	<p>Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX5            01 : Select .1 input for MUX5            10 : Select .2 input for MUX5            11 : Select .3 input for MUX5</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX4            01 : Select .1 input for MUX4            10 : Select .2 input for MUX4            11 : Select .3 input for MUX4</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX3            01 : Select .1 input for MUX3            10 : Select .2 input for MUX3            11 : Select .3 input for MUX3</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX2            01 : Select .1 input for MUX2            10 : Select .2 input for MUX2            11 : Select .3 input for MUX2</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-97. AUXSIG0MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	<p>Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX1  01 : Select .1 input for MUX1  10 : Select .2 input for MUX1  11 : Select .3 input for MUX1</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX0  01 : Select .1 input for MUX0  10 : Select .2 input for MUX0  11 : Select .3 input for MUX0</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.2 AUXSIG0MUX16TO31CFG Register (Offset = 2h) [reset = 0h]

AUXSIG0MUX16TO31CFG is shown in [Figure 9-88](#) and described in [Table 9-98](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-0

**Figure 9-88. AUXSIG0MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-98. AUXSIG0MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG0 of CLB-XBAR  00 : Select .0 input for MUX31 01 : Select .1 input for MUX31 10 : Select .2 input for MUX31 11 : Select .3 input for MUX31  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG0 of CLB-XBAR  00 : Select .0 input for MUX30 01 : Select .1 input for MUX30 10 : Select .2 input for MUX30 11 : Select .3 input for MUX30  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG0 of CLB-XBAR  00 : Select .0 input for MUX29 01 : Select .1 input for MUX29 10 : Select .2 input for MUX29 11 : Select .3 input for MUX29  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG0 of CLB-XBAR  00 : Select .0 input for MUX28 01 : Select .1 input for MUX28 10 : Select .2 input for MUX28 11 : Select .3 input for MUX28  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-98. AUXSIG0MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX27  01 : Select .1 input for MUX27  10 : Select .2 input for MUX27  11 : Select .3 input for MUX27</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX26  01 : Select .1 input for MUX26  10 : Select .2 input for MUX26  11 : Select .3 input for MUX26</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX25  01 : Select .1 input for MUX25  10 : Select .2 input for MUX25  11 : Select .3 input for MUX25</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX24  01 : Select .1 input for MUX24  10 : Select .2 input for MUX24  11 : Select .3 input for MUX24</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX23  01 : Select .1 input for MUX23  10 : Select .2 input for MUX23  11 : Select .3 input for MUX23</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



**Table 9-98. AUXSIG0MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX22	R/W	0h	<p>Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX22            01 : Select .1 input for MUX22            10 : Select .2 input for MUX22            11 : Select .3 input for MUX22</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX21	R/W	0h	<p>Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX21            01 : Select .1 input for MUX21            10 : Select .2 input for MUX21            11 : Select .3 input for MUX21</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX20            01 : Select .1 input for MUX20            10 : Select .2 input for MUX20            11 : Select .3 input for MUX20</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX19            01 : Select .1 input for MUX19            10 : Select .2 input for MUX19            11 : Select .3 input for MUX19</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX18            01 : Select .1 input for MUX18            10 : Select .2 input for MUX18            11 : Select .3 input for MUX18</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-98. AUXSIG0MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	<p>Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX17  01 : Select .1 input for MUX17  10 : Select .2 input for MUX17  11 : Select .3 input for MUX17</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG0 of CLB-XBAR</p> <p>00 : Select .0 input for MUX16  01 : Select .1 input for MUX16  10 : Select .2 input for MUX16  11 : Select .3 input for MUX16</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.3 AUXSIG1MUX0TO15CFG Register (Offset = 4h) [reset = 0h]

AUXSIG1MUX0TO15CFG is shown in [Figure 9-89](#) and described in [Table 9-99](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-1

**Figure 9-89. AUXSIG1MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-99. AUXSIG1MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG1 of CLB-XBAR  00 : Select .0 input for MUX15 01 : Select .1 input for MUX15 10 : Select .2 input for MUX15 11 : Select .3 input for MUX15  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG1 of CLB-XBAR  00 : Select .0 input for MUX14 01 : Select .1 input for MUX14 10 : Select .2 input for MUX14 11 : Select .3 input for MUX14  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG1 of CLB-XBAR  00 : Select .0 input for MUX13 01 : Select .1 input for MUX13 10 : Select .2 input for MUX13 11 : Select .3 input for MUX13  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG1 of CLB-XBAR  00 : Select .0 input for MUX12 01 : Select .1 input for MUX12 10 : Select .2 input for MUX12 11 : Select .3 input for MUX12  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-99. AUXSIG1MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX11  01 : Select .1 input for MUX11  10 : Select .2 input for MUX11  11 : Select .3 input for MUX11</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX10  01 : Select .1 input for MUX10  10 : Select .2 input for MUX10  11 : Select .3 input for MUX10</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX9  01 : Select .1 input for MUX9  10 : Select .2 input for MUX9  11 : Select .3 input for MUX9</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX8  01 : Select .1 input for MUX8  10 : Select .2 input for MUX8  11 : Select .3 input for MUX8</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX7  01 : Select .1 input for MUX7  10 : Select .2 input for MUX7  11 : Select .3 input for MUX7</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-99. AUXSIG1MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX6	R/W	0h	<p>Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX6            01 : Select .1 input for MUX6            10 : Select .2 input for MUX6            11 : Select .3 input for MUX6</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX5	R/W	0h	<p>Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX5            01 : Select .1 input for MUX5            10 : Select .2 input for MUX5            11 : Select .3 input for MUX5</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX4            01 : Select .1 input for MUX4            10 : Select .2 input for MUX4            11 : Select .3 input for MUX4</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX3            01 : Select .1 input for MUX3            10 : Select .2 input for MUX3            11 : Select .3 input for MUX3</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX2            01 : Select .1 input for MUX2            10 : Select .2 input for MUX2            11 : Select .3 input for MUX2</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-99. AUXSIG1MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	<p>Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX1  01 : Select .1 input for MUX1  10 : Select .2 input for MUX1  11 : Select .3 input for MUX1</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX0  01 : Select .1 input for MUX0  10 : Select .2 input for MUX0  11 : Select .3 input for MUX0</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.4 AUXSIG1MUX16TO31CFG Register (Offset = 6h) [reset = 0h]

AUXSIG1MUX16TO31CFG is shown in [Figure 9-90](#) and described in [Table 9-100](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-1

**Figure 9-90. AUXSIG1MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-100. AUXSIG1MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG1 of CLB-XBAR  00 : Select .0 input for MUX31 01 : Select .1 input for MUX31 10 : Select .2 input for MUX31 11 : Select .3 input for MUX31  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG1 of CLB-XBAR  00 : Select .0 input for MUX30 01 : Select .1 input for MUX30 10 : Select .2 input for MUX30 11 : Select .3 input for MUX30  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG1 of CLB-XBAR  00 : Select .0 input for MUX29 01 : Select .1 input for MUX29 10 : Select .2 input for MUX29 11 : Select .3 input for MUX29  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG1 of CLB-XBAR  00 : Select .0 input for MUX28 01 : Select .1 input for MUX28 10 : Select .2 input for MUX28 11 : Select .3 input for MUX28  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-100. AUXSIG1MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX27  01 : Select .1 input for MUX27  10 : Select .2 input for MUX27  11 : Select .3 input for MUX27</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX26  01 : Select .1 input for MUX26  10 : Select .2 input for MUX26  11 : Select .3 input for MUX26</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX25  01 : Select .1 input for MUX25  10 : Select .2 input for MUX25  11 : Select .3 input for MUX25</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX24  01 : Select .1 input for MUX24  10 : Select .2 input for MUX24  11 : Select .3 input for MUX24</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX23  01 : Select .1 input for MUX23  10 : Select .2 input for MUX23  11 : Select .3 input for MUX23</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



**Table 9-100. AUXSIG1MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX22	R/W	0h	<p>Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX22            01 : Select .1 input for MUX22            10 : Select .2 input for MUX22            11 : Select .3 input for MUX22</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX21	R/W	0h	<p>Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX21            01 : Select .1 input for MUX21            10 : Select .2 input for MUX21            11 : Select .3 input for MUX21</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX20            01 : Select .1 input for MUX20            10 : Select .2 input for MUX20            11 : Select .3 input for MUX20</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX19            01 : Select .1 input for MUX19            10 : Select .2 input for MUX19            11 : Select .3 input for MUX19</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX18            01 : Select .1 input for MUX18            10 : Select .2 input for MUX18            11 : Select .3 input for MUX18</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-100. AUXSIG1MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	<p>Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX17  01 : Select .1 input for MUX17  10 : Select .2 input for MUX17  11 : Select .3 input for MUX17</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG1 of CLB-XBAR</p> <p>00 : Select .0 input for MUX16  01 : Select .1 input for MUX16  10 : Select .2 input for MUX16  11 : Select .3 input for MUX16</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.5 AUXSIG2MUX0TO15CFG Register (Offset = 8h) [reset = 0h]

AUXSIG2MUX0TO15CFG is shown in [Figure 9-91](#) and described in [Table 9-101](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-2

**Figure 9-91. AUXSIG2MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-101. AUXSIG2MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG2 of CLB-XBAR  00 : Select .0 input for MUX15 01 : Select .1 input for MUX15 10 : Select .2 input for MUX15 11 : Select .3 input for MUX15  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG2 of CLB-XBAR  00 : Select .0 input for MUX14 01 : Select .1 input for MUX14 10 : Select .2 input for MUX14 11 : Select .3 input for MUX14  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG2 of CLB-XBAR  00 : Select .0 input for MUX13 01 : Select .1 input for MUX13 10 : Select .2 input for MUX13 11 : Select .3 input for MUX13  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG2 of CLB-XBAR  00 : Select .0 input for MUX12 01 : Select .1 input for MUX12 10 : Select .2 input for MUX12 11 : Select .3 input for MUX12  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-101. AUXSIG2MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX11  01 : Select .1 input for MUX11  10 : Select .2 input for MUX11  11 : Select .3 input for MUX11</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX10  01 : Select .1 input for MUX10  10 : Select .2 input for MUX10  11 : Select .3 input for MUX10</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX9  01 : Select .1 input for MUX9  10 : Select .2 input for MUX9  11 : Select .3 input for MUX9</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX8  01 : Select .1 input for MUX8  10 : Select .2 input for MUX8  11 : Select .3 input for MUX8</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX7  01 : Select .1 input for MUX7  10 : Select .2 input for MUX7  11 : Select .3 input for MUX7</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-101. AUXSIG2MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX6	R/W	0h	<p>Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX6            01 : Select .1 input for MUX6            10 : Select .2 input for MUX6            11 : Select .3 input for MUX6</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX5	R/W	0h	<p>Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX5            01 : Select .1 input for MUX5            10 : Select .2 input for MUX5            11 : Select .3 input for MUX5</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX4            01 : Select .1 input for MUX4            10 : Select .2 input for MUX4            11 : Select .3 input for MUX4</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX3            01 : Select .1 input for MUX3            10 : Select .2 input for MUX3            11 : Select .3 input for MUX3</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX2            01 : Select .1 input for MUX2            10 : Select .2 input for MUX2            11 : Select .3 input for MUX2</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-101. AUXSIG2MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	<p>Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX1  01 : Select .1 input for MUX1  10 : Select .2 input for MUX1  11 : Select .3 input for MUX1</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX0  01 : Select .1 input for MUX0  10 : Select .2 input for MUX0  11 : Select .3 input for MUX0</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.6 AUXSIG2MUX16TO31CFG Register (Offset = Ah) [reset = 0h]

AUXSIG2MUX16TO31CFG is shown in [Figure 9-92](#) and described in [Table 9-102](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-2

**Figure 9-92. AUXSIG2MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-102. AUXSIG2MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG2 of CLB-XBAR  00 : Select .0 input for MUX31 01 : Select .1 input for MUX31 10 : Select .2 input for MUX31 11 : Select .3 input for MUX31  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG2 of CLB-XBAR  00 : Select .0 input for MUX30 01 : Select .1 input for MUX30 10 : Select .2 input for MUX30 11 : Select .3 input for MUX30  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG2 of CLB-XBAR  00 : Select .0 input for MUX29 01 : Select .1 input for MUX29 10 : Select .2 input for MUX29 11 : Select .3 input for MUX29  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG2 of CLB-XBAR  00 : Select .0 input for MUX28 01 : Select .1 input for MUX28 10 : Select .2 input for MUX28 11 : Select .3 input for MUX28  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-102. AUXSIG2MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX27  01 : Select .1 input for MUX27  10 : Select .2 input for MUX27  11 : Select .3 input for MUX27</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX26  01 : Select .1 input for MUX26  10 : Select .2 input for MUX26  11 : Select .3 input for MUX26</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX25  01 : Select .1 input for MUX25  10 : Select .2 input for MUX25  11 : Select .3 input for MUX25</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX24  01 : Select .1 input for MUX24  10 : Select .2 input for MUX24  11 : Select .3 input for MUX24</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX23  01 : Select .1 input for MUX23  10 : Select .2 input for MUX23  11 : Select .3 input for MUX23</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



**Table 9-102. AUXSIG2MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX22	R/W	0h	<p>Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX22            01 : Select .1 input for MUX22            10 : Select .2 input for MUX22            11 : Select .3 input for MUX22</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX21	R/W	0h	<p>Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX21            01 : Select .1 input for MUX21            10 : Select .2 input for MUX21            11 : Select .3 input for MUX21</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX20            01 : Select .1 input for MUX20            10 : Select .2 input for MUX20            11 : Select .3 input for MUX20</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX19            01 : Select .1 input for MUX19            10 : Select .2 input for MUX19            11 : Select .3 input for MUX19</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX18            01 : Select .1 input for MUX18            10 : Select .2 input for MUX18            11 : Select .3 input for MUX18</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-102. AUXSIG2MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	<p>Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX17  01 : Select .1 input for MUX17  10 : Select .2 input for MUX17  11 : Select .3 input for MUX17</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG2 of CLB-XBAR</p> <p>00 : Select .0 input for MUX16  01 : Select .1 input for MUX16  10 : Select .2 input for MUX16  11 : Select .3 input for MUX16</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.7 AUXSIG3MUX0TO15CFG Register (Offset = Ch) [reset = 0h]

AUXSIG3MUX0TO15CFG is shown in [Figure 9-93](#) and described in [Table 9-103](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-3

**Figure 9-93. AUXSIG3MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-103. AUXSIG3MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG3 of CLB-XBAR  00 : Select .0 input for MUX15 01 : Select .1 input for MUX15 10 : Select .2 input for MUX15 11 : Select .3 input for MUX15  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG3 of CLB-XBAR  00 : Select .0 input for MUX14 01 : Select .1 input for MUX14 10 : Select .2 input for MUX14 11 : Select .3 input for MUX14  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG3 of CLB-XBAR  00 : Select .0 input for MUX13 01 : Select .1 input for MUX13 10 : Select .2 input for MUX13 11 : Select .3 input for MUX13  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG3 of CLB-XBAR  00 : Select .0 input for MUX12 01 : Select .1 input for MUX12 10 : Select .2 input for MUX12 11 : Select .3 input for MUX12  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-103. AUXSIG3MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX11  01 : Select .1 input for MUX11  10 : Select .2 input for MUX11  11 : Select .3 input for MUX11</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX10  01 : Select .1 input for MUX10  10 : Select .2 input for MUX10  11 : Select .3 input for MUX10</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX9  01 : Select .1 input for MUX9  10 : Select .2 input for MUX9  11 : Select .3 input for MUX9</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX8  01 : Select .1 input for MUX8  10 : Select .2 input for MUX8  11 : Select .3 input for MUX8</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX7  01 : Select .1 input for MUX7  10 : Select .2 input for MUX7  11 : Select .3 input for MUX7</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-103. AUXSIG3MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX6	R/W	0h	<p>Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX6            01 : Select .1 input for MUX6            10 : Select .2 input for MUX6            11 : Select .3 input for MUX6</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX5	R/W	0h	<p>Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX5            01 : Select .1 input for MUX5            10 : Select .2 input for MUX5            11 : Select .3 input for MUX5</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX4            01 : Select .1 input for MUX4            10 : Select .2 input for MUX4            11 : Select .3 input for MUX4</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX3            01 : Select .1 input for MUX3            10 : Select .2 input for MUX3            11 : Select .3 input for MUX3</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX2            01 : Select .1 input for MUX2            10 : Select .2 input for MUX2            11 : Select .3 input for MUX2</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-103. AUXSIG3MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	<p>Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX1            01 : Select .1 input for MUX1            10 : Select .2 input for MUX1            11 : Select .3 input for MUX1</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX0            01 : Select .1 input for MUX0            10 : Select .2 input for MUX0            11 : Select .3 input for MUX0</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**9.3.6.8 AUXSIG3MUX16TO31CFG Register (Offset = Eh) [reset = 0h]**

AUXSIG3MUX16TO31CFG is shown in [Figure 9-94](#) and described in [Table 9-104](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-3

**Figure 9-94. AUXSIG3MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-104. AUXSIG3MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG3 of CLB-XBAR  00 : Select .0 input for MUX31 01 : Select .1 input for MUX31 10 : Select .2 input for MUX31 11 : Select .3 input for MUX31  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG3 of CLB-XBAR  00 : Select .0 input for MUX30 01 : Select .1 input for MUX30 10 : Select .2 input for MUX30 11 : Select .3 input for MUX30  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG3 of CLB-XBAR  00 : Select .0 input for MUX29 01 : Select .1 input for MUX29 10 : Select .2 input for MUX29 11 : Select .3 input for MUX29  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG3 of CLB-XBAR  00 : Select .0 input for MUX28 01 : Select .1 input for MUX28 10 : Select .2 input for MUX28 11 : Select .3 input for MUX28  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-104. AUXSIG3MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX27  01 : Select .1 input for MUX27  10 : Select .2 input for MUX27  11 : Select .3 input for MUX27</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX26  01 : Select .1 input for MUX26  10 : Select .2 input for MUX26  11 : Select .3 input for MUX26</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX25  01 : Select .1 input for MUX25  10 : Select .2 input for MUX25  11 : Select .3 input for MUX25</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX24  01 : Select .1 input for MUX24  10 : Select .2 input for MUX24  11 : Select .3 input for MUX24</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX23  01 : Select .1 input for MUX23  10 : Select .2 input for MUX23  11 : Select .3 input for MUX23</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



**Table 9-104. AUXSIG3MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX22	R/W	0h	<p>Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX22            01 : Select .1 input for MUX22            10 : Select .2 input for MUX22            11 : Select .3 input for MUX22</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX21	R/W	0h	<p>Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX21            01 : Select .1 input for MUX21            10 : Select .2 input for MUX21            11 : Select .3 input for MUX21</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX20            01 : Select .1 input for MUX20            10 : Select .2 input for MUX20            11 : Select .3 input for MUX20</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX19            01 : Select .1 input for MUX19            10 : Select .2 input for MUX19            11 : Select .3 input for MUX19</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX18            01 : Select .1 input for MUX18            10 : Select .2 input for MUX18            11 : Select .3 input for MUX18</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-104. AUXSIG3MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	<p>Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX17  01 : Select .1 input for MUX17  10 : Select .2 input for MUX17  11 : Select .3 input for MUX17</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG3 of CLB-XBAR</p> <p>00 : Select .0 input for MUX16  01 : Select .1 input for MUX16  10 : Select .2 input for MUX16  11 : Select .3 input for MUX16</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.9 AUXSIG4MUX0TO15CFG Register (Offset = 10h) [reset = 0h]

AUXSIG4MUX0TO15CFG is shown in [Figure 9-95](#) and described in [Table 9-105](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-4

**Figure 9-95. AUXSIG4MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-105. AUXSIG4MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG4 of CLB-XBAR  00 : Select .0 input for MUX15 01 : Select .1 input for MUX15 10 : Select .2 input for MUX15 11 : Select .3 input for MUX15  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG4 of CLB-XBAR  00 : Select .0 input for MUX14 01 : Select .1 input for MUX14 10 : Select .2 input for MUX14 11 : Select .3 input for MUX14  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG4 of CLB-XBAR  00 : Select .0 input for MUX13 01 : Select .1 input for MUX13 10 : Select .2 input for MUX13 11 : Select .3 input for MUX13  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG4 of CLB-XBAR  00 : Select .0 input for MUX12 01 : Select .1 input for MUX12 10 : Select .2 input for MUX12 11 : Select .3 input for MUX12  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-105. AUXSIG4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX11  01 : Select .1 input for MUX11  10 : Select .2 input for MUX11  11 : Select .3 input for MUX11</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX10  01 : Select .1 input for MUX10  10 : Select .2 input for MUX10  11 : Select .3 input for MUX10</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX9  01 : Select .1 input for MUX9  10 : Select .2 input for MUX9  11 : Select .3 input for MUX9</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX8  01 : Select .1 input for MUX8  10 : Select .2 input for MUX8  11 : Select .3 input for MUX8</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX7  01 : Select .1 input for MUX7  10 : Select .2 input for MUX7  11 : Select .3 input for MUX7</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-105. AUXSIG4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX6	R/W	0h	<p>Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX6            01 : Select .1 input for MUX6            10 : Select .2 input for MUX6            11 : Select .3 input for MUX6</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX5	R/W	0h	<p>Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX5            01 : Select .1 input for MUX5            10 : Select .2 input for MUX5            11 : Select .3 input for MUX5</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX4            01 : Select .1 input for MUX4            10 : Select .2 input for MUX4            11 : Select .3 input for MUX4</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX3            01 : Select .1 input for MUX3            10 : Select .2 input for MUX3            11 : Select .3 input for MUX3</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX2            01 : Select .1 input for MUX2            10 : Select .2 input for MUX2            11 : Select .3 input for MUX2</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-105. AUXSIG4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	<p>Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX1  01 : Select .1 input for MUX1  10 : Select .2 input for MUX1  11 : Select .3 input for MUX1</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX0  01 : Select .1 input for MUX0  10 : Select .2 input for MUX0  11 : Select .3 input for MUX0</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.10 AUXSIG4MUX16TO31CFG Register (Offset = 12h) [reset = 0h]

AUXSIG4MUX16TO31CFG is shown in [Figure 9-96](#) and described in [Table 9-106](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-4

**Figure 9-96. AUXSIG4MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-106. AUXSIG4MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG4 of CLB-XBAR  00 : Select .0 input for MUX31 01 : Select .1 input for MUX31 10 : Select .2 input for MUX31 11 : Select .3 input for MUX31  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG4 of CLB-XBAR  00 : Select .0 input for MUX30 01 : Select .1 input for MUX30 10 : Select .2 input for MUX30 11 : Select .3 input for MUX30  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG4 of CLB-XBAR  00 : Select .0 input for MUX29 01 : Select .1 input for MUX29 10 : Select .2 input for MUX29 11 : Select .3 input for MUX29  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG4 of CLB-XBAR  00 : Select .0 input for MUX28 01 : Select .1 input for MUX28 10 : Select .2 input for MUX28 11 : Select .3 input for MUX28  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-106. AUXSIG4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX27  01 : Select .1 input for MUX27  10 : Select .2 input for MUX27  11 : Select .3 input for MUX27</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX26  01 : Select .1 input for MUX26  10 : Select .2 input for MUX26  11 : Select .3 input for MUX26</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX25  01 : Select .1 input for MUX25  10 : Select .2 input for MUX25  11 : Select .3 input for MUX25</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX24  01 : Select .1 input for MUX24  10 : Select .2 input for MUX24  11 : Select .3 input for MUX24</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX23  01 : Select .1 input for MUX23  10 : Select .2 input for MUX23  11 : Select .3 input for MUX23</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



**Table 9-106. AUXSIG4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX22	R/W	0h	<p>Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX22            01 : Select .1 input for MUX22            10 : Select .2 input for MUX22            11 : Select .3 input for MUX22</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX21	R/W	0h	<p>Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX21            01 : Select .1 input for MUX21            10 : Select .2 input for MUX21            11 : Select .3 input for MUX21</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX20            01 : Select .1 input for MUX20            10 : Select .2 input for MUX20            11 : Select .3 input for MUX20</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX19            01 : Select .1 input for MUX19            10 : Select .2 input for MUX19            11 : Select .3 input for MUX19</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX18            01 : Select .1 input for MUX18            10 : Select .2 input for MUX18            11 : Select .3 input for MUX18</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-106. AUXSIG4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	<p>Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX17  01 : Select .1 input for MUX17  10 : Select .2 input for MUX17  11 : Select .3 input for MUX17</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG4 of CLB-XBAR</p> <p>00 : Select .0 input for MUX16  01 : Select .1 input for MUX16  10 : Select .2 input for MUX16  11 : Select .3 input for MUX16</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.11 AUXSIG5MUX0TO15CFG Register (Offset = 14h) [reset = 0h]

AUXSIG5MUX0TO15CFG is shown in [Figure 9-97](#) and described in [Table 9-107](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-5

**Figure 9-97. AUXSIG5MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-107. AUXSIG5MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG5 of CLB-XBAR  00 : Select .0 input for MUX15 01 : Select .1 input for MUX15 10 : Select .2 input for MUX15 11 : Select .3 input for MUX15  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG5 of CLB-XBAR  00 : Select .0 input for MUX14 01 : Select .1 input for MUX14 10 : Select .2 input for MUX14 11 : Select .3 input for MUX14  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG5 of CLB-XBAR  00 : Select .0 input for MUX13 01 : Select .1 input for MUX13 10 : Select .2 input for MUX13 11 : Select .3 input for MUX13  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG5 of CLB-XBAR  00 : Select .0 input for MUX12 01 : Select .1 input for MUX12 10 : Select .2 input for MUX12 11 : Select .3 input for MUX12  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-107. AUXSIG5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX11  01 : Select .1 input for MUX11  10 : Select .2 input for MUX11  11 : Select .3 input for MUX11</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX10  01 : Select .1 input for MUX10  10 : Select .2 input for MUX10  11 : Select .3 input for MUX10</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX9  01 : Select .1 input for MUX9  10 : Select .2 input for MUX9  11 : Select .3 input for MUX9</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX8  01 : Select .1 input for MUX8  10 : Select .2 input for MUX8  11 : Select .3 input for MUX8</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX7  01 : Select .1 input for MUX7  10 : Select .2 input for MUX7  11 : Select .3 input for MUX7</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-107. AUXSIG5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX6	R/W	0h	<p>Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX6            01 : Select .1 input for MUX6            10 : Select .2 input for MUX6            11 : Select .3 input for MUX6</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX5	R/W	0h	<p>Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX5            01 : Select .1 input for MUX5            10 : Select .2 input for MUX5            11 : Select .3 input for MUX5</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX4            01 : Select .1 input for MUX4            10 : Select .2 input for MUX4            11 : Select .3 input for MUX4</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX3            01 : Select .1 input for MUX3            10 : Select .2 input for MUX3            11 : Select .3 input for MUX3</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX2            01 : Select .1 input for MUX2            10 : Select .2 input for MUX2            11 : Select .3 input for MUX2</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-107. AUXSIG5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	<p>Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX1  01 : Select .1 input for MUX1  10 : Select .2 input for MUX1  11 : Select .3 input for MUX1</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX0  01 : Select .1 input for MUX0  10 : Select .2 input for MUX0  11 : Select .3 input for MUX0</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.12 AUXSIG5MUX16TO31CFG Register (Offset = 16h) [reset = 0h]

AUXSIG5MUX16TO31CFG is shown in [Figure 9-98](#) and described in [Table 9-108](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-5

**Figure 9-98. AUXSIG5MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-108. AUXSIG5MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG5 of CLB-XBAR  00 : Select .0 input for MUX31 01 : Select .1 input for MUX31 10 : Select .2 input for MUX31 11 : Select .3 input for MUX31  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG5 of CLB-XBAR  00 : Select .0 input for MUX30 01 : Select .1 input for MUX30 10 : Select .2 input for MUX30 11 : Select .3 input for MUX30  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG5 of CLB-XBAR  00 : Select .0 input for MUX29 01 : Select .1 input for MUX29 10 : Select .2 input for MUX29 11 : Select .3 input for MUX29  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG5 of CLB-XBAR  00 : Select .0 input for MUX28 01 : Select .1 input for MUX28 10 : Select .2 input for MUX28 11 : Select .3 input for MUX28  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-108. AUXSIG5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX27  01 : Select .1 input for MUX27  10 : Select .2 input for MUX27  11 : Select .3 input for MUX27</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX26  01 : Select .1 input for MUX26  10 : Select .2 input for MUX26  11 : Select .3 input for MUX26</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX25  01 : Select .1 input for MUX25  10 : Select .2 input for MUX25  11 : Select .3 input for MUX25</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX24  01 : Select .1 input for MUX24  10 : Select .2 input for MUX24  11 : Select .3 input for MUX24</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX23  01 : Select .1 input for MUX23  10 : Select .2 input for MUX23  11 : Select .3 input for MUX23</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



**Table 9-108. AUXSIG5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX22	R/W	0h	<p>Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX22            01 : Select .1 input for MUX22            10 : Select .2 input for MUX22            11 : Select .3 input for MUX22</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX21	R/W	0h	<p>Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX21            01 : Select .1 input for MUX21            10 : Select .2 input for MUX21            11 : Select .3 input for MUX21</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX20            01 : Select .1 input for MUX20            10 : Select .2 input for MUX20            11 : Select .3 input for MUX20</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX19            01 : Select .1 input for MUX19            10 : Select .2 input for MUX19            11 : Select .3 input for MUX19</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX18            01 : Select .1 input for MUX18            10 : Select .2 input for MUX18            11 : Select .3 input for MUX18</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-108. AUXSIG5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	<p>Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX17  01 : Select .1 input for MUX17  10 : Select .2 input for MUX17  11 : Select .3 input for MUX17</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG5 of CLB-XBAR</p> <p>00 : Select .0 input for MUX16  01 : Select .1 input for MUX16  10 : Select .2 input for MUX16  11 : Select .3 input for MUX16</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.13 AUXSIG6MUX0TO15CFG Register (Offset = 18h) [reset = 0h]

AUXSIG6MUX0TO15CFG is shown in [Figure 9-99](#) and described in [Table 9-109](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-6

**Figure 9-99. AUXSIG6MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-109. AUXSIG6MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG6 of CLB-XBAR  00 : Select .0 input for MUX15 01 : Select .1 input for MUX15 10 : Select .2 input for MUX15 11 : Select .3 input for MUX15  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG6 of CLB-XBAR  00 : Select .0 input for MUX14 01 : Select .1 input for MUX14 10 : Select .2 input for MUX14 11 : Select .3 input for MUX14  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG6 of CLB-XBAR  00 : Select .0 input for MUX13 01 : Select .1 input for MUX13 10 : Select .2 input for MUX13 11 : Select .3 input for MUX13  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG6 of CLB-XBAR  00 : Select .0 input for MUX12 01 : Select .1 input for MUX12 10 : Select .2 input for MUX12 11 : Select .3 input for MUX12  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-109. AUXSIG6MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX11  01 : Select .1 input for MUX11  10 : Select .2 input for MUX11  11 : Select .3 input for MUX11</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX10  01 : Select .1 input for MUX10  10 : Select .2 input for MUX10  11 : Select .3 input for MUX10</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX9  01 : Select .1 input for MUX9  10 : Select .2 input for MUX9  11 : Select .3 input for MUX9</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX8  01 : Select .1 input for MUX8  10 : Select .2 input for MUX8  11 : Select .3 input for MUX8</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX7  01 : Select .1 input for MUX7  10 : Select .2 input for MUX7  11 : Select .3 input for MUX7</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-109. AUXSIG6MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX6	R/W	0h	<p>Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX6            01 : Select .1 input for MUX6            10 : Select .2 input for MUX6            11 : Select .3 input for MUX6</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX5	R/W	0h	<p>Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX5            01 : Select .1 input for MUX5            10 : Select .2 input for MUX5            11 : Select .3 input for MUX5</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX4            01 : Select .1 input for MUX4            10 : Select .2 input for MUX4            11 : Select .3 input for MUX4</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX3            01 : Select .1 input for MUX3            10 : Select .2 input for MUX3            11 : Select .3 input for MUX3</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX2            01 : Select .1 input for MUX2            10 : Select .2 input for MUX2            11 : Select .3 input for MUX2</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-109. AUXSIG6MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	<p>Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX1  01 : Select .1 input for MUX1  10 : Select .2 input for MUX1  11 : Select .3 input for MUX1</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX0  01 : Select .1 input for MUX0  10 : Select .2 input for MUX0  11 : Select .3 input for MUX0</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

### 9.3.6.14 AUXSIG6MUX16TO31CFG Register (Offset = 1Ah) [reset = 0h]

AUXSIG6MUX16TO31CFG is shown in [Figure 9-100](#) and described in [Table 9-110](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-6

**Figure 9-100. AUXSIG6MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-110. AUXSIG6MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG6 of CLB-XBAR  00 : Select .0 input for MUX31 01 : Select .1 input for MUX31 10 : Select .2 input for MUX31 11 : Select .3 input for MUX31  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG6 of CLB-XBAR  00 : Select .0 input for MUX30 01 : Select .1 input for MUX30 10 : Select .2 input for MUX30 11 : Select .3 input for MUX30  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG6 of CLB-XBAR  00 : Select .0 input for MUX29 01 : Select .1 input for MUX29 10 : Select .2 input for MUX29 11 : Select .3 input for MUX29  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG6 of CLB-XBAR  00 : Select .0 input for MUX28 01 : Select .1 input for MUX28 10 : Select .2 input for MUX28 11 : Select .3 input for MUX28  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-110. AUXSIG6MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX27  01 : Select .1 input for MUX27  10 : Select .2 input for MUX27  11 : Select .3 input for MUX27</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX26  01 : Select .1 input for MUX26  10 : Select .2 input for MUX26  11 : Select .3 input for MUX26</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX25  01 : Select .1 input for MUX25  10 : Select .2 input for MUX25  11 : Select .3 input for MUX25</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX24  01 : Select .1 input for MUX24  10 : Select .2 input for MUX24  11 : Select .3 input for MUX24</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX23  01 : Select .1 input for MUX23  10 : Select .2 input for MUX23  11 : Select .3 input for MUX23</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



**Table 9-110. AUXSIG6MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX22	R/W	0h	<p>Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX22            01 : Select .1 input for MUX22            10 : Select .2 input for MUX22            11 : Select .3 input for MUX22</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX21	R/W	0h	<p>Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX21            01 : Select .1 input for MUX21            10 : Select .2 input for MUX21            11 : Select .3 input for MUX21</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX20            01 : Select .1 input for MUX20            10 : Select .2 input for MUX20            11 : Select .3 input for MUX20</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX19            01 : Select .1 input for MUX19            10 : Select .2 input for MUX19            11 : Select .3 input for MUX19</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX18            01 : Select .1 input for MUX18            10 : Select .2 input for MUX18            11 : Select .3 input for MUX18</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-110. AUXSIG6MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	<p>Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX17            01 : Select .1 input for MUX17            10 : Select .2 input for MUX17            11 : Select .3 input for MUX17</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG6 of CLB-XBAR</p> <p>00 : Select .0 input for MUX16            01 : Select .1 input for MUX16            10 : Select .2 input for MUX16            11 : Select .3 input for MUX16</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

### 9.3.6.15 AUXSIG7MUX0TO15CFG Register (Offset = 1Ch) [reset = 0h]

AUXSIG7MUX0TO15CFG is shown in [Figure 9-101](#) and described in [Table 9-111](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-7

**Figure 9-101. AUXSIG7MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
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	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
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 9-111. AUXSIG7MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for MUX15: Selects 4X1 group output for AUXSIG7 of CLB-XBAR  00 : Select .0 input for MUX15 01 : Select .1 input for MUX15 10 : Select .2 input for MUX15 11 : Select .3 input for MUX15  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Group Select Bits for MUX14: Selects 4X1 group output for AUXSIG7 of CLB-XBAR  00 : Select .0 input for MUX14 01 : Select .1 input for MUX14 10 : Select .2 input for MUX14 11 : Select .3 input for MUX14  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Group Select Bits for MUX13: Selects 4X1 group output for AUXSIG7 of CLB-XBAR  00 : Select .0 input for MUX13 01 : Select .1 input for MUX13 10 : Select .2 input for MUX13 11 : Select .3 input for MUX13  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Group Select Bits for MUX12: Selects 4X1 group output for AUXSIG7 of CLB-XBAR  00 : Select .0 input for MUX12 01 : Select .1 input for MUX12 10 : Select .2 input for MUX12 11 : Select .3 input for MUX12  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-111. AUXSIG7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX11	R/W	0h	<p>Group Select Bits for MUX11: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX11  01 : Select .1 input for MUX11  10 : Select .2 input for MUX11  11 : Select .3 input for MUX11</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX10	R/W	0h	<p>Group Select Bits for MUX10: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX10  01 : Select .1 input for MUX10  10 : Select .2 input for MUX10  11 : Select .3 input for MUX10</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX9	R/W	0h	<p>Group Select Bits for MUX9: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX9  01 : Select .1 input for MUX9  10 : Select .2 input for MUX9  11 : Select .3 input for MUX9</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX8	R/W	0h	<p>Group Select Bits for MUX8: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX8  01 : Select .1 input for MUX8  10 : Select .2 input for MUX8  11 : Select .3 input for MUX8</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX7	R/W	0h	<p>Group Select Bits for MUX7: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX7  01 : Select .1 input for MUX7  10 : Select .2 input for MUX7  11 : Select .3 input for MUX7</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>

**Table 9-111. AUXSIG7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX6	R/W	0h	<p>Group Select Bits for MUX6: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX6            01 : Select .1 input for MUX6            10 : Select .2 input for MUX6            11 : Select .3 input for MUX6</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX5	R/W	0h	<p>Group Select Bits for MUX5: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX5            01 : Select .1 input for MUX5            10 : Select .2 input for MUX5            11 : Select .3 input for MUX5</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX4	R/W	0h	<p>Group Select Bits for MUX4: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX4            01 : Select .1 input for MUX4            10 : Select .2 input for MUX4            11 : Select .3 input for MUX4</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX3	R/W	0h	<p>Group Select Bits for MUX3: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX3            01 : Select .1 input for MUX3            10 : Select .2 input for MUX3            11 : Select .3 input for MUX3</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX2	R/W	0h	<p>Group Select Bits for MUX2: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX2            01 : Select .1 input for MUX2            10 : Select .2 input for MUX2            11 : Select .3 input for MUX2</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-111. AUXSIG7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	<p>Group Select Bits for MUX1: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX1            01 : Select .1 input for MUX1            10 : Select .2 input for MUX1            11 : Select .3 input for MUX1</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
1-0	MUX0	R/W	0h	<p>Group Select Bits for MUX0: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX0            01 : Select .1 input for MUX0            10 : Select .2 input for MUX0            11 : Select .3 input for MUX0</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

### 9.3.6.16 AUXSIG7MUX16TO31CFG Register (Offset = 1Eh) [reset = 0h]

AUXSIG7MUX16TO31CFG is shown in [Figure 9-102](#) and described in [Table 9-112](#).

Return to the [Summary Table](#).

CLB XBAR Mux Configuration for Output-7

**Figure 9-102. AUXSIG7MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
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	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
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 9-112. AUXSIG7MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for MUX31: Selects 4X1 group output for AUXSIG7 of CLB-XBAR  00 : Select .0 input for MUX31 01 : Select .1 input for MUX31 10 : Select .2 input for MUX31 11 : Select .3 input for MUX31  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Group Select Bits for MUX30: Selects 4X1 group output for AUXSIG7 of CLB-XBAR  00 : Select .0 input for MUX30 01 : Select .1 input for MUX30 10 : Select .2 input for MUX30 11 : Select .3 input for MUX30  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Group Select Bits for MUX29: Selects 4X1 group output for AUXSIG7 of CLB-XBAR  00 : Select .0 input for MUX29 01 : Select .1 input for MUX29 10 : Select .2 input for MUX29 11 : Select .3 input for MUX29  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Group Select Bits for MUX28: Selects 4X1 group output for AUXSIG7 of CLB-XBAR  00 : Select .0 input for MUX28 01 : Select .1 input for MUX28 10 : Select .2 input for MUX28 11 : Select .3 input for MUX28  Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-112. AUXSIG7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	MUX27	R/W	0h	<p>Group Select Bits for MUX27: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX27  01 : Select .1 input for MUX27  10 : Select .2 input for MUX27  11 : Select .3 input for MUX27</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
21-20	MUX26	R/W	0h	<p>Group Select Bits for MUX26: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX26  01 : Select .1 input for MUX26  10 : Select .2 input for MUX26  11 : Select .3 input for MUX26</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
19-18	MUX25	R/W	0h	<p>Group Select Bits for MUX25: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX25  01 : Select .1 input for MUX25  10 : Select .2 input for MUX25  11 : Select .3 input for MUX25</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
17-16	MUX24	R/W	0h	<p>Group Select Bits for MUX24: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX24  01 : Select .1 input for MUX24  10 : Select .2 input for MUX24  11 : Select .3 input for MUX24</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>
15-14	MUX23	R/W	0h	<p>Group Select Bits for MUX23: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX23  01 : Select .1 input for MUX23  10 : Select .2 input for MUX23  11 : Select .3 input for MUX23</p> <p>Refer to the CLB X-BAR section of this chapter for more details.  Reset type: CPU1.SYSRSn</p>



**Table 9-112. AUXSIG7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	MUX22	R/W	0h	<p>Group Select Bits for MUX22: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX22            01 : Select .1 input for MUX22            10 : Select .2 input for MUX22            11 : Select .3 input for MUX22</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
11-10	MUX21	R/W	0h	<p>Group Select Bits for MUX21: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX21            01 : Select .1 input for MUX21            10 : Select .2 input for MUX21            11 : Select .3 input for MUX21</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
9-8	MUX20	R/W	0h	<p>Group Select Bits for MUX20: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX20            01 : Select .1 input for MUX20            10 : Select .2 input for MUX20            11 : Select .3 input for MUX20</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
7-6	MUX19	R/W	0h	<p>Group Select Bits for MUX19: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX19            01 : Select .1 input for MUX19            10 : Select .2 input for MUX19            11 : Select .3 input for MUX19</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
5-4	MUX18	R/W	0h	<p>Group Select Bits for MUX18: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX18            01 : Select .1 input for MUX18            10 : Select .2 input for MUX18            11 : Select .3 input for MUX18</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

**Table 9-112. AUXSIG7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	<p>Group Select Bits for MUX17: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX17            01 : Select .1 input for MUX17            10 : Select .2 input for MUX17            11 : Select .3 input for MUX17</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>
1-0	MUX16	R/W	0h	<p>Group Select Bits for MUX16: Selects 4X1 group output for AUXSIG7 of CLB-XBAR</p> <p>00 : Select .0 input for MUX16            01 : Select .1 input for MUX16            10 : Select .2 input for MUX16            11 : Select .3 input for MUX16</p> <p>Refer to the CLB X-BAR section of this chapter for more details.            Reset type: CPU1.SYSRSn</p>

### 9.3.6.17 AUXSIG0MUXENABLE Register (Offset = 20h) [reset = 0h]

AUXSIG0MUXENABLE is shown in [Figure 9-103](#) and described in [Table 9-113](#).

Return to the [Summary Table](#).

CLB XBAR Mux Enable Register for Output-0

**Figure 9-103. AUXSIG0MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-113. AUXSIG0MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of MUX31 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX31 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX31 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of MUX30 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX30 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX30 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of MUX29 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX29 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX29 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of MUX28 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX28 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX28 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-113. AUXSIG0MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of MUX27 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX27 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX27 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of MUX26 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX26 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX26 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of MUX25 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX25 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX25 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of MUX24 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX24 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX24 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of MUX23 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX23 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX23 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of MUX22 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX22 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX22 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of MUX21 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX21 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX21 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-113. AUXSIG0MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of MUX20 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX20 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX20 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of MUX19 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX19 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX19 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of MUX18 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX18 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX18 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of MUX17 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX17 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX17 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of MUX16 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX16 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX16 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of MUX15 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX15 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX15 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of MUX14 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX14 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX14 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-113. AUXSIG0MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of MUX13 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX13 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX13 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of MUX12 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX12 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX12 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of MUX11 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX11 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX11 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of MUX10 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX10 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX10 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of MUX9 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX9 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX9 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of MUX8 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX8 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX8 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of MUX7 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX7 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX7 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-113. AUXSIG0MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of MUX6 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX6 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX6 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of MUX5 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX5 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX5 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of MUX4 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX4 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX4 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of MUX3 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX3 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX3 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of MUX2 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX2 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX2 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of MUX1 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX1 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX1 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive AUXSIG0 of CLB-XBAR</p> <p>0: Respective output of MUX0 is disabled to drive the AUXSIG0 of CLB-XBAR</p> <p>1: Respective output of MUX0 is enabled to drive the AUXSIG0 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.6.18 AUXSIG1MUXENABLE Register (Offset = 22h) [reset = 0h]

AUXSIG1MUXENABLE is shown in [Figure 9-104](#) and described in [Table 9-114](#).

Return to the [Summary Table](#).

CLB XBAR Mux Enable Register for Output-1

**Figure 9-104. AUXSIG1MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-114. AUXSIG1MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of MUX31 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX31 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX31 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of MUX30 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX30 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX30 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of MUX29 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX29 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX29 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of MUX28 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX28 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX28 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>



**Table 9-114. AUXSIG1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of MUX27 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX27 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX27 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of MUX26 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX26 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX26 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of MUX25 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX25 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX25 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of MUX24 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX24 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX24 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of MUX23 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX23 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX23 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of MUX22 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX22 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX22 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of MUX21 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX21 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX21 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-114. AUXSIG1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of MUX20 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX20 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX20 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of MUX19 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX19 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX19 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of MUX18 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX18 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX18 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of MUX17 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX17 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX17 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of MUX16 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX16 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX16 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of MUX15 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX15 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX15 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of MUX14 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX14 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX14 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-114. AUXSIG1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of MUX13 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX13 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX13 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of MUX12 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX12 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX12 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of MUX11 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX11 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX11 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of MUX10 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX10 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX10 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of MUX9 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX9 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX9 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of MUX8 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX8 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX8 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of MUX7 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX7 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX7 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-114. AUXSIG1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of MUX6 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX6 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX6 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of MUX5 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX5 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX5 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of MUX4 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX4 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX4 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of MUX3 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX3 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX3 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of MUX2 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX2 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX2 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of MUX1 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX1 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX1 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive AUXSIG1 of CLB-XBAR</p> <p>0: Respective output of MUX0 is disabled to drive the AUXSIG1 of CLB-XBAR</p> <p>1: Respective output of MUX0 is enabled to drive the AUXSIG1 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.6.19 AUXSIG2MUXENABLE Register (Offset = 24h) [reset = 0h]

AUXSIG2MUXENABLE is shown in [Figure 9-105](#) and described in [Table 9-115](#).

Return to the [Summary Table](#).

CLB XBAR Mux Enable Register for Output-2

**Figure 9-105. AUXSIG2MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-115. AUXSIG2MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of MUX31 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX31 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX31 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of MUX30 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX30 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX30 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of MUX29 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX29 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX29 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of MUX28 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX28 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX28 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-115. AUXSIG2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of MUX27 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX27 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX27 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of MUX26 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX26 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX26 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of MUX25 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX25 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX25 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of MUX24 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX24 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX24 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of MUX23 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX23 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX23 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of MUX22 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX22 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX22 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of MUX21 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX21 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX21 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-115. AUXSIG2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of MUX20 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX20 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX20 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of MUX19 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX19 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX19 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of MUX18 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX18 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX18 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of MUX17 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX17 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX17 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of MUX16 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX16 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX16 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of MUX15 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX15 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX15 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of MUX14 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX14 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX14 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-115. AUXSIG2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of MUX13 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX13 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX13 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of MUX12 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX12 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX12 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of MUX11 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX11 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX11 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of MUX10 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX10 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX10 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of MUX9 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX9 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX9 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of MUX8 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX8 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX8 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of MUX7 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX7 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX7 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>



**Table 9-115. AUXSIG2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of MUX6 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX6 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX6 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of MUX5 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX5 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX5 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of MUX4 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX4 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX4 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of MUX3 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX3 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX3 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of MUX2 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX2 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX2 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of MUX1 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX1 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX1 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive AUXSIG2 of CLB-XBAR</p> <p>0: Respective output of MUX0 is disabled to drive the AUXSIG2 of CLB-XBAR</p> <p>1: Respective output of MUX0 is enabled to drive the AUXSIG2 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.6.20 AUXSIG3MUXENABLE Register (Offset = 26h) [reset = 0h]

AUXSIG3MUXENABLE is shown in [Figure 9-106](#) and described in [Table 9-116](#).

Return to the [Summary Table](#).

CLB XBAR Mux Enable Register for Output-3

**Figure 9-106. AUXSIG3MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-116. AUXSIG3MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of MUX31 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX31 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX31 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of MUX30 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX30 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX30 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of MUX29 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX29 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX29 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of MUX28 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX28 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX28 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-116. AUXSIG3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of MUX27 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX27 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX27 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of MUX26 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX26 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX26 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of MUX25 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX25 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX25 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of MUX24 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX24 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX24 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of MUX23 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX23 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX23 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of MUX22 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX22 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX22 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of MUX21 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX21 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX21 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-116. AUXSIG3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of MUX20 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX20 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX20 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of MUX19 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX19 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX19 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of MUX18 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX18 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX18 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of MUX17 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX17 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX17 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of MUX16 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX16 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX16 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of MUX15 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX15 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX15 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of MUX14 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX14 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX14 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-116. AUXSIG3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of MUX13 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX13 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX13 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of MUX12 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX12 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX12 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of MUX11 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX11 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX11 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of MUX10 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX10 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX10 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of MUX9 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX9 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX9 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of MUX8 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX8 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX8 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of MUX7 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX7 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX7 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-116. AUXSIG3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of MUX6 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX6 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX6 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of MUX5 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX5 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX5 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of MUX4 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX4 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX4 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of MUX3 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX3 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX3 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of MUX2 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX2 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX2 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of MUX1 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX1 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX1 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive AUXSIG3 of CLB-XBAR</p> <p>0: Respective output of MUX0 is disabled to drive the AUXSIG3 of CLB-XBAR</p> <p>1: Respective output of MUX0 is enabled to drive the AUXSIG3 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.6.21 AUXSIG4MUXENABLE Register (Offset = 28h) [reset = 0h]

AUXSIG4MUXENABLE is shown in [Figure 9-107](#) and described in [Table 9-117](#).

Return to the [Summary Table](#).

CLB XBAR Mux Enable Register for Output-4

**Figure 9-107. AUXSIG4MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
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
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
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
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
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 9-117. AUXSIG4MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of MUX31 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX31 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX31 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of MUX30 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX30 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX30 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of MUX29 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX29 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX29 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of MUX28 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX28 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX28 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-117. AUXSIG4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of MUX27 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX27 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX27 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of MUX26 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX26 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX26 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of MUX25 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX25 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX25 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of MUX24 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX24 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX24 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of MUX23 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX23 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX23 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of MUX22 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX22 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX22 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of MUX21 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX21 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX21 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>



**Table 9-117. AUXSIG4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of MUX20 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX20 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX20 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of MUX19 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX19 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX19 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of MUX18 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX18 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX18 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of MUX17 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX17 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX17 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of MUX16 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX16 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX16 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of MUX15 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX15 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX15 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of MUX14 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX14 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX14 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-117. AUXSIG4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of MUX13 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX13 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX13 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of MUX12 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX12 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX12 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of MUX11 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX11 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX11 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of MUX10 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX10 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX10 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of MUX9 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX9 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX9 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of MUX8 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX8 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX8 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of MUX7 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX7 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX7 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-117. AUXSIG4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of MUX6 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX6 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX6 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of MUX5 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX5 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX5 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of MUX4 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX4 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX4 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of MUX3 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX3 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX3 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of MUX2 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX2 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX2 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of MUX1 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX1 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX1 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive AUXSIG4 of CLB-XBAR</p> <p>0: Respective output of MUX0 is disabled to drive the AUXSIG4 of CLB-XBAR</p> <p>1: Respective output of MUX0 is enabled to drive the AUXSIG4 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.6.22 AUXSIG5MUXENABLE Register (Offset = 2Ah) [reset = 0h]

AUXSIG5MUXENABLE is shown in [Figure 9-108](#) and described in [Table 9-118](#).

Return to the [Summary Table](#).

CLB XBAR Mux Enable Register for Output-5

**Figure 9-108. AUXSIG5MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-118. AUXSIG5MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of MUX31 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX31 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX31 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of MUX30 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX30 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX30 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of MUX29 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX29 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX29 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of MUX28 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX28 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX28 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-118. AUXSIG5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of MUX27 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX27 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX27 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of MUX26 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX26 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX26 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of MUX25 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX25 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX25 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of MUX24 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX24 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX24 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of MUX23 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX23 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX23 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of MUX22 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX22 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX22 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of MUX21 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX21 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX21 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-118. AUXSIG5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of MUX20 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX20 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX20 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of MUX19 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX19 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX19 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of MUX18 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX18 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX18 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of MUX17 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX17 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX17 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of MUX16 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX16 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX16 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of MUX15 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX15 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX15 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of MUX14 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX14 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX14 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-118. AUXSIG5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of MUX13 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX13 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX13 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of MUX12 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX12 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX12 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of MUX11 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX11 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX11 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of MUX10 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX10 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX10 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of MUX9 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX9 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX9 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of MUX8 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX8 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX8 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of MUX7 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX7 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX7 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-118. AUXSIG5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of MUX6 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX6 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX6 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of MUX5 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX5 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX5 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of MUX4 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX4 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX4 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of MUX3 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX3 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX3 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of MUX2 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX2 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX2 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of MUX1 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX1 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX1 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive AUXSIG5 of CLB-XBAR</p> <p>0: Respective output of MUX0 is disabled to drive the AUXSIG5 of CLB-XBAR</p> <p>1: Respective output of MUX0 is enabled to drive the AUXSIG5 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>



### 9.3.6.23 AUXSIG6MUXENABLE Register (Offset = 2Ch) [reset = 0h]

AUXSIG6MUXENABLE is shown in [Figure 9-109](#) and described in [Table 9-119](#).

Return to the [Summary Table](#).

CLB XBAR Mux Enable Register for Output-6

**Figure 9-109. AUXSIG6MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-119. AUXSIG6MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of MUX31 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX31 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX31 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of MUX30 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX30 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX30 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of MUX29 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX29 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX29 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of MUX28 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX28 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX28 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-119. AUXSIG6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of MUX27 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX27 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX27 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of MUX26 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX26 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX26 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of MUX25 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX25 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX25 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of MUX24 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX24 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX24 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of MUX23 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX23 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX23 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of MUX22 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX22 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX22 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of MUX21 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX21 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX21 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-119. AUXSIG6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of MUX20 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX20 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX20 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of MUX19 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX19 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX19 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of MUX18 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX18 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX18 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of MUX17 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX17 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX17 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of MUX16 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX16 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX16 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of MUX15 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX15 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX15 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of MUX14 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX14 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX14 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-119. AUXSIG6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of MUX13 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX13 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX13 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of MUX12 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX12 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX12 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of MUX11 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX11 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX11 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of MUX10 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX10 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX10 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of MUX9 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX9 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX9 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of MUX8 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX8 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX8 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of MUX7 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX7 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX7 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-119. AUXSIG6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of MUX6 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX6 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX6 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of MUX5 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX5 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX5 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of MUX4 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX4 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX4 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of MUX3 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX3 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX3 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of MUX2 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX2 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX2 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of MUX1 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX1 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX1 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive AUXSIG6 of CLB-XBAR</p> <p>0: Respective output of MUX0 is disabled to drive the AUXSIG6 of CLB-XBAR</p> <p>1: Respective output of MUX0 is enabled to drive the AUXSIG6 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

### 9.3.6.24 AUXSIG7MUXENABLE Register (Offset = 2Eh) [reset = 0h]

AUXSIG7MUXENABLE is shown in [Figure 9-110](#) and described in [Table 9-120](#).

Return to the [Summary Table](#).

CLB XBAR Mux Enable Register for Output-7

**Figure 9-110. AUXSIG7MUXENABLE Register**

31		30		29		28		27		26		25		24	
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24	MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23		22		21		20		19		18		17		16	
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16	MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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	
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	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 9-120. AUXSIG7MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	<p>Selects the output of MUX31 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX31 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX31 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
30	MUX30	R/W	0h	<p>Selects the output of MUX30 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX30 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX30 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
29	MUX29	R/W	0h	<p>Selects the output of MUX29 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX29 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX29 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
28	MUX28	R/W	0h	<p>Selects the output of MUX28 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX28 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX28 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-120. AUXSIG7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	<p>Selects the output of MUX27 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX27 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX27 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
26	MUX26	R/W	0h	<p>Selects the output of MUX26 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX26 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX26 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
25	MUX25	R/W	0h	<p>Selects the output of MUX25 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX25 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX25 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
24	MUX24	R/W	0h	<p>Selects the output of MUX24 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX24 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX24 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
23	MUX23	R/W	0h	<p>Selects the output of MUX23 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX23 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX23 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
22	MUX22	R/W	0h	<p>Selects the output of MUX22 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX22 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX22 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
21	MUX21	R/W	0h	<p>Selects the output of MUX21 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX21 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX21 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>

**Table 9-120. AUXSIG7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	MUX20	R/W	0h	<p>Selects the output of MUX20 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX20 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX20 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
19	MUX19	R/W	0h	<p>Selects the output of MUX19 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX19 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX19 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
18	MUX18	R/W	0h	<p>Selects the output of MUX18 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX18 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX18 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
17	MUX17	R/W	0h	<p>Selects the output of MUX17 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX17 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX17 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
16	MUX16	R/W	0h	<p>Selects the output of MUX16 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX16 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX16 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
15	MUX15	R/W	0h	<p>Selects the output of MUX15 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX15 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX15 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>
14	MUX14	R/W	0h	<p>Selects the output of MUX14 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX14 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX14 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details.</p> <p>Reset type: CPU1.SYSRSn</p>



**Table 9-120. AUXSIG7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	MUX13	R/W	0h	<p>Selects the output of MUX13 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX13 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX13 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
12	MUX12	R/W	0h	<p>Selects the output of MUX12 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX12 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX12 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
11	MUX11	R/W	0h	<p>Selects the output of MUX11 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX11 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX11 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
10	MUX10	R/W	0h	<p>Selects the output of MUX10 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX10 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX10 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
9	MUX9	R/W	0h	<p>Selects the output of MUX9 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX9 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX9 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
8	MUX8	R/W	0h	<p>Selects the output of MUX8 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX8 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX8 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
7	MUX7	R/W	0h	<p>Selects the output of MUX7 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX7 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX7 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**Table 9-120. AUXSIG7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	MUX6	R/W	0h	<p>Selects the output of MUX6 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX6 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX6 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
5	MUX5	R/W	0h	<p>Selects the output of MUX5 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX5 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX5 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
4	MUX4	R/W	0h	<p>Selects the output of MUX4 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX4 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX4 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
3	MUX3	R/W	0h	<p>Selects the output of MUX3 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX3 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX3 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	MUX2	R/W	0h	<p>Selects the output of MUX2 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX2 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX2 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	MUX1	R/W	0h	<p>Selects the output of MUX1 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX1 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX1 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	MUX0	R/W	0h	<p>Selects the output of mux0 to drive AUXSIG7 of CLB-XBAR</p> <p>0: Respective output of MUX0 is disabled to drive the AUXSIG7 of CLB-XBAR</p> <p>1: Respective output of MUX0 is enabled to drive the AUXSIG7 of CLB-XBAR</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**9.3.6.25 AUXSIGOUTINV Register (Offset = 38h) [reset = 0h]**

AUXSIGOUTINV is shown in [Figure 9-111](#) and described in [Table 9-121](#).

Return to the [Summary Table](#).

CLB XBAR Output Inversion Register

**Figure 9-111. AUXSIGOUTINV Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
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 9-121. AUXSIGOUTINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUT7	R/W	0h	Selects polarity for AUXSIG7 of CLB-XBAR 0: drives active high output 1: drives active-low output Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	OUT6	R/W	0h	Selects polarity for AUXSIG6 of CLB-XBAR 0: drives active high output 1: drives active-low output Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	OUT5	R/W	0h	Selects polarity for AUXSIG5 of CLB-XBAR 0: drives active high output 1: drives active-low output Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	OUT4	R/W	0h	Selects polarity for AUXSIG4 of CLB-XBAR 0: drives active high output 1: drives active-low output Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-121. AUXSIGOUTINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	OUT3	R/W	0h	<p>Selects polarity for AUXSIG3 of CLB-XBAR</p> <p>0: drives active high output 1: drives active-low output</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
2	OUT2	R/W	0h	<p>Selects polarity for AUXSIG2 of CLB-XBAR</p> <p>0: drives active high output 1: drives active-low output</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
1	OUT1	R/W	0h	<p>Selects polarity for AUXSIG1 of CLB-XBAR</p> <p>0: drives active high output 1: drives active-low output</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>
0	OUT0	R/W	0h	<p>Selects polarity for AUXSIG0 of CLB-XBAR</p> <p>0: drives active high output 1: drives active-low output</p> <p>Refer to the CLB X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn</p>

**9.3.6.26 AUXSIGLOCK Register (Offset = 3Eh) [reset = 0h]**

AUXSIGLOCK is shown in [Figure 9-112](#) and described in [Table 9-122](#).

Return to the [Summary Table](#).

ClbXbar Configuration Lock register

**Figure 9-112. AUXSIGLOCK Register**

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

**Table 9-122. AUXSIGLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	Bit-0 of this register can be set only if KEY= 0x5a5a Reset type: CPU1.SYSRSn
15-1	RESERVED	R-0	0h	Reserved
0	LOCK	R/WOnce	0h	Locks the configuration for CLB-XBAR. Once the configuration is locked, writes to the below registers for CLB-XBAR is blocked.  Registers Affected by the LOCK mechanism: CLB-XBAROUTyMUX0TO15CFG CLB-XBAROUTyMUX16TO31CFG CLB-XBAROUTyMUXENABLE CLB-XBAROUTLATEN CLB-XBAROUTINV  0: Writes to the above registers are allowed 1: Writes to the above registers are blocked  Note: [1] LOCK mechanism only applies to writes. Reads are never blocked.  Reset type: CPU1.SYSRSn

### 9.3.7 Register to Driverlib Function Mapping

**Table 9-123. EPWMXBAR Registers to Driverlib Functions**

File	Driverlib Function
<b>TRIP4MUX0TO15CFG</b>	
xbar.c	XBAR_setEPWMMuxConfig
<b>TRIP4MUX16TO31CFG</b>	
xbar.c	XBAR_setEPWMMuxConfig
<b>TRIP5MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP5MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP7MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP7MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP8MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP8MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP9MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP9MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP10MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP10MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP11MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP11MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP12MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP12MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP4MUXENABLE</b>	
xbar.h	XBAR_enableEPWMMux
xbar.h	XBAR_disableEPWMMux
<b>TRIP5MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP7MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP8MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP9MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP10MUXENABLE</b>	
-	See TRIP4MUXENABLE

**Table 9-123. EPWMXBAR Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>TRIP11MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP12MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIPOUTINV</b>	
xbar.h	XBAR_invertEPWMSignal
<b>TRIPLOCK</b>	
xbar.h	XBAR_lockEPWM

**Table 9-124. INPUTXBAR Registers to Driverlib Functions**

File	Driverlib Function
<b>INPUT1SELECT</b>	
xbar.h	XBAR_setInputPin
<b>INPUT2SELECT</b>	
-	See INPUT1SELECT
<b>INPUT3SELECT</b>	
-	See INPUT1SELECT
<b>INPUT4SELECT</b>	
-	See INPUT1SELECT
<b>INPUT5SELECT</b>	
-	See INPUT1SELECT
<b>INPUT6SELECT</b>	
-	See INPUT1SELECT
<b>INPUT7SELECT</b>	
-	See INPUT1SELECT
<b>INPUT8SELECT</b>	
-	See INPUT1SELECT
<b>INPUT9SELECT</b>	
-	See INPUT1SELECT
<b>INPUT10SELECT</b>	
-	See INPUT1SELECT
<b>INPUT11SELECT</b>	
-	See INPUT1SELECT
<b>INPUT12SELECT</b>	
-	See INPUT1SELECT
<b>INPUT13SELECT</b>	
-	See INPUT1SELECT
<b>INPUT14SELECT</b>	
-	See INPUT1SELECT
<b>INPUT15SELECT</b>	
-	See INPUT1SELECT
<b>INPUT16SELECT</b>	
-	See INPUT1SELECT
<b>INPUTSELECTLOCK</b>	
xbar.h	XBAR_lockInput

**Table 9-125. OUTPUTXBAR Registers to Driverlib Functions**

File	Driverlib Function
<b>OUTPUT1MUX0TO15CFG</b>	
xbar.c	XBAR_setOutputMuxConfig
<b>OUTPUT1MUX16TO31CFG</b>	
xbar.c	XBAR_setOutputMuxConfig
<b>OUTPUT2MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT2MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT3MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT3MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT4MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT4MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT5MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT5MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT6MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT6MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT7MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT7MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT8MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT8MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT1MUXENABLE</b>	
xbar.h	XBAR_enableOutputMux
xbar.h	XBAR_disableOutputMux
<b>OUTPUT2MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT3MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT4MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT5MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT6MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT7MUXENABLE</b>	
-	See OUTPUT1MUXENABLE



**Table 9-125. OUTPUTXBAR Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>OUTPUT8MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUTLATCH</b>	
xbar.h	XBAR_setOutputLatchMode
xbar.h	XBAR_getOutputLatchStatus
xbar.h	XBAR_clearOutputLatch
xbar.h	XBAR_forceOutputLatch
<b>OUTPUTLATCHCLR</b>	
xbar.h	XBAR_clearOutputLatch
<b>OUTPUTLATCHFRC</b>	
xbar.h	XBAR_forceOutputLatch
<b>OUTPUTLATCHENABLE</b>	
xbar.h	XBAR_setOutputLatchMode
<b>OUTPUTINV</b>	
xbar.h	XBAR_invertOutputSignal
<b>OUTPUTLOCK</b>	
xbar.h	XBAR_lockOutput

**Table 9-126. XBAR Registers to Driverlib Functions**

File	Driverlib Function
<b>FLG1</b>	
xbar.c	XBAR_getInputFlagStatus
<b>FLG2</b>	
xbar.c	XBAR_getInputFlagStatus
<b>FLG3</b>	
xbar.c	XBAR_getInputFlagStatus
<b>FLG4</b>	
xbar.c	XBAR_getInputFlagStatus
<b>CLR1</b>	
xbar.c	XBAR_clearInputFlag
<b>CLR2</b>	
xbar.c	XBAR_clearInputFlag
<b>CLR3</b>	
xbar.c	XBAR_clearInputFlag
<b>CLR4</b>	
xbar.c	XBAR_clearInputFlag

**Table 9-127. CLB XBAR Registers to Driverlib Functions**

File	Driverlib Function
<b>AUXSIG0MUX0TO15CFG</b>	
xbar.c	XBAR_setCLBMuxConfig
<b>AUXSIG1MUX0TO15CFG</b>	
-	See AUXSIG0MUX0TO15CFG
<b>AUXSIG2MUX0TO15CFG</b>	
-	See AUXSIG0MUX0TO15CFG

**Table 9-127. CLBxBAR Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>AUXSIG3MUX0TO15CFG</b>	
-	See AUXSIG0MUX0TO15CFG
<b>AUXSIG4MUX0TO15CFG</b>	
-	See AUXSIG0MUX0TO15CFG
<b>AUXSIG5MUX0TO15CFG</b>	
-	See AUXSIG0MUX0TO15CFG
<b>AUXSIG6MUX0TO15CFG</b>	
-	See AUXSIG0MUX0TO15CFG
<b>AUXSIG7MUX0TO15CFG</b>	
-	See AUXSIG0MUX0TO15CFG
<b>AUXSIG0MUXENABLE</b>	
xbar.h	XBAR_enableCLBMux
xbar.h	XBAR_disableCLBMux
<b>AUXSIG1MUXENABLE</b>	
-	See AUXSIG0MUXENABLE
<b>AUXSIG2MUXENABLE</b>	
-	See AUXSIG0MUXENABLE
<b>AUXSIG3MUXENABLE</b>	
-	See AUXSIG0MUXENABLE
<b>AUXSIG4MUXENABLE</b>	
-	See AUXSIG0MUXENABLE
<b>AUXSIG5MUXENABLE</b>	
-	See AUXSIG0MUXENABLE
<b>AUXSIG6MUXENABLE</b>	
-	See AUXSIG0MUXENABLE
<b>AUXSIG7MUXENABLE</b>	
-	See AUXSIG0MUXENABLE
<b>AUXSIGOUTINV</b>	
xbar.h	XBAR_invertCLBSignal

## ***Direct Memory Access (DMA)***

The direct memory access (DMA) module provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. Additionally, the DMA has the capability to orthogonally rearrange the data as it is transferred as well as “ping-pong” data between buffers. These features are useful for structuring data into blocks for optimal CPU processing.

Topic	Page
<b>10.1 Introduction .....</b>	<b>1364</b>
<b>10.2 Features .....</b>	<b>1364</b>
<b>10.3 Architecture .....</b>	<b>1365</b>
<b>10.4 Address Pointer and Transfer Control .....</b>	<b>1369</b>
<b>10.5 Pipeline Timing and Throughput .....</b>	<b>1374</b>
<b>10.6 CPU and CLA Arbitration .....</b>	<b>1375</b>
<b>10.7 Channel Priority .....</b>	<b>1376</b>
<b>10.8 Overrun Detection Feature .....</b>	<b>1377</b>
<b>10.9 DMA Registers .....</b>	<b>1378</b>

## 10.1 Introduction

The strength of a controller is not measured purely in processor speed, but in total system capabilities. As a part of the equation, any time the CPU bandwidth for a given function can be reduced, the greater the system capabilities. Many times applications spend a significant amount of their bandwidth moving data, whether it is from off-chip memory to on-chip memory, or from a peripheral such as an analog-to-digital converter (ADC) to RAM, or even from one peripheral to another. Furthermore, many times this data comes in a format that is not conducive to the optimum processing powers of the CPU. The DMA module described in this reference guide has the ability to free up CPU bandwidth and rearrange the data into a pattern for more streamlined processing.

The DMA module is an event-based machine, meaning it requires a peripheral or software trigger to start a DMA transfer. Although it can be made into a periodic time-driven machine by configuring a timer as the DMA trigger source, there is no mechanism within the module itself to start memory transfers periodically. The DMA module has six independent DMA channels which can be configured separately and each channel contains its own independent PIE interrupt to let the CPU know when a DMA transfer has either started or completed. Five of the six channels are exactly the same, while Channel 1 has the ability to be configured at a higher priority than the others. At the heart of the DMA is a state machine and tightly coupled address control logic. It is this address control logic that allows for rearrangement of the block of data during the transfer as well as the process of ping-ponging data between buffers. Each of these features, along with others, will be discussed in detail in this document.

## 10.2 Features

DMA features include:

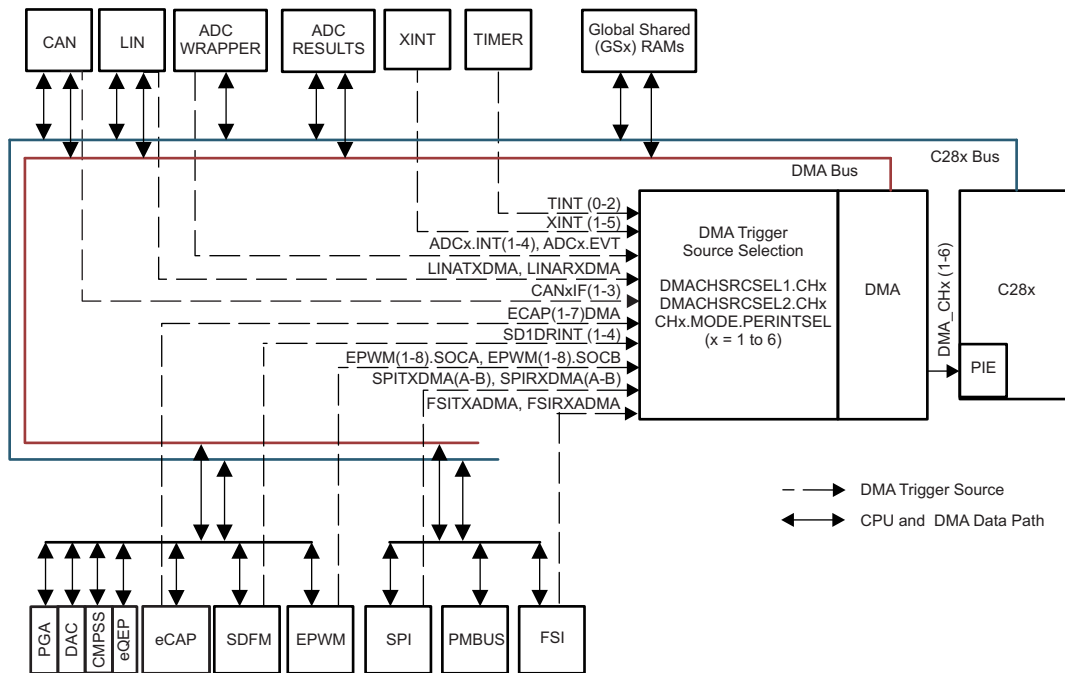
- Six channels with independent PIE interrupts
- Each DMA channel can be triggered from multiple peripheral trigger sources independently
- Word Size: 16-bit or 32-bit (SPI limited to 16-bit)
- Throughput: 3 cycles/word without arbitration

### 10.3 Architecture

#### 10.3.1 Block Diagram

Figure 10-1 shows a device-level block diagram of the DMA.

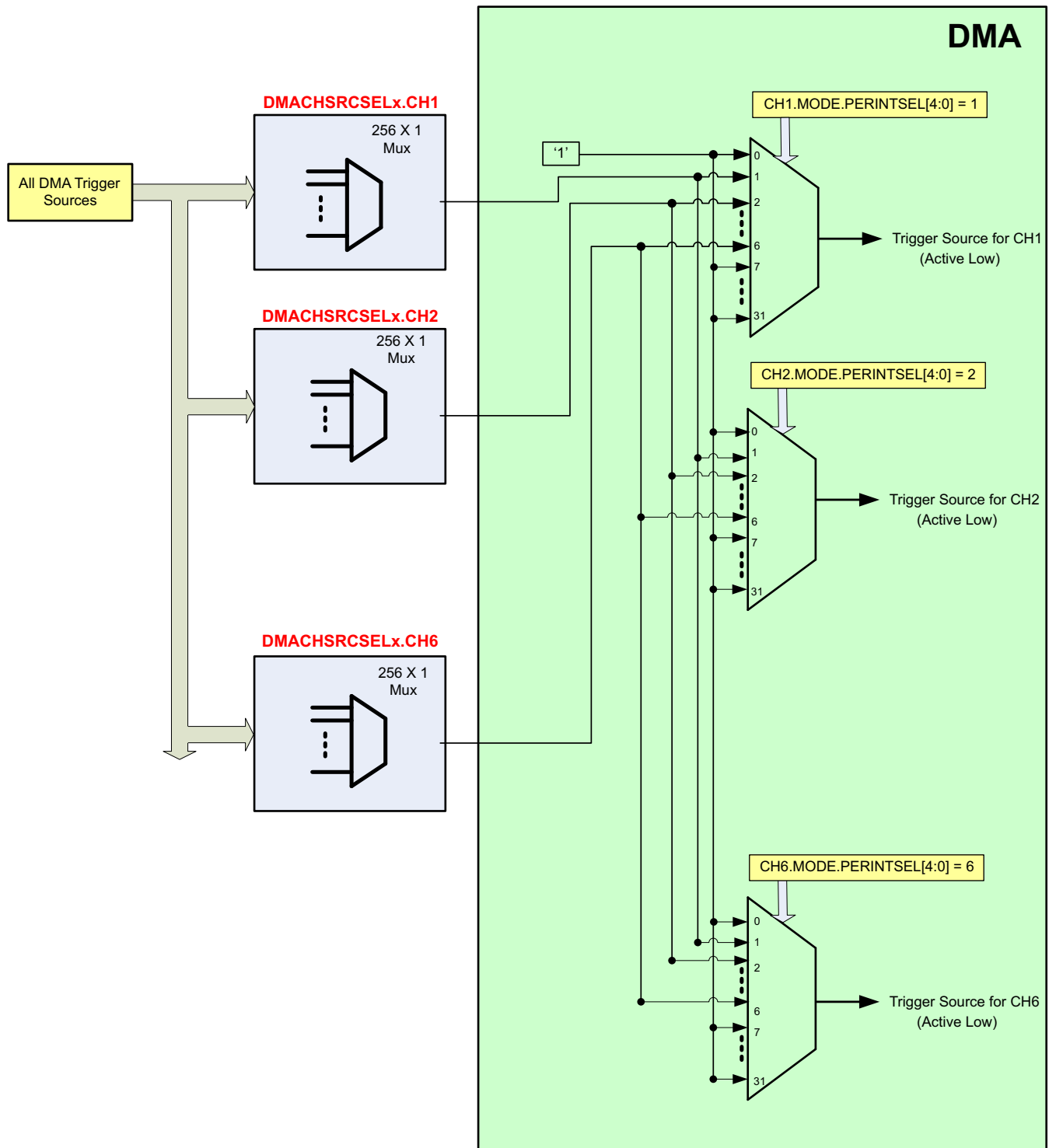
Figure 10-1. DMA Block Diagram



#### 10.3.2 Peripheral Interrupt Event Trigger Sources

Each DMA Channel can be configured to trigger by software and other peripheral triggers events. DMACHSRCSELx register can be used to configure DMA Trigger sources for each DMA channel. CHx.MODE.PERINTSEL register bitfield should be set to channel number (CHx.MODE.PERINTSEL = x) as shown in Figure 10-2. Included in these DMA Trigger sources are five external interrupt signals which can be connected to most of the general-purpose input/output (GPIO) pins on the device. This adds significant flexibility to the event trigger capabilities. Upon receipt of a peripheral interrupt event signal, the DMA will automatically send a clear signal to the interrupt source so that subsequent interrupt events will occur.

Figure 10-2. DMA Trigger Architecture



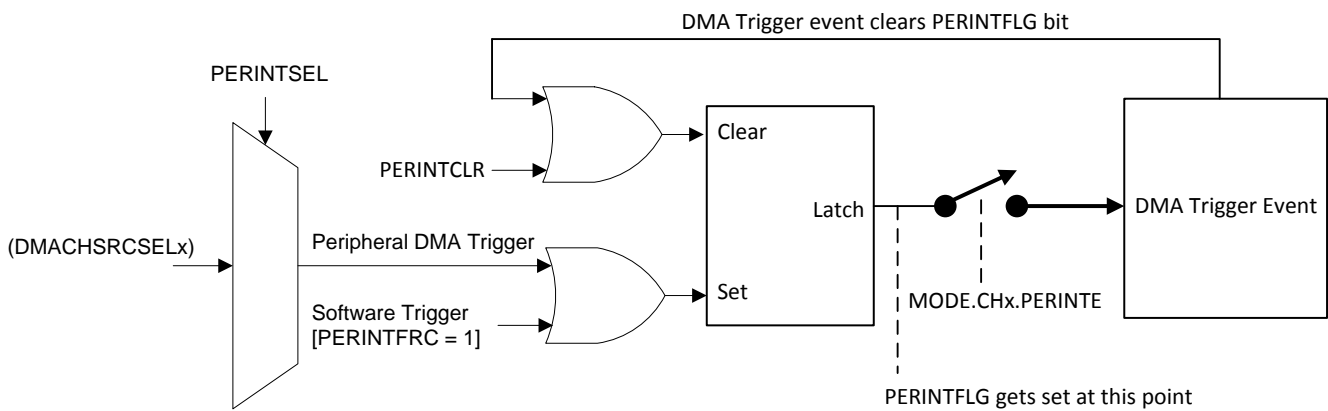
**NOTE:** To use the system level DMA Trigger source selection, the DMA internal trigger source selection configuration for each channel should be done using the DMACHSRCSELx register, and the CHx.MODE.PERINTSEL register as shown here. See Table 10-1 or the DMACHSRCSELx register definition for a complete list of DMA trigger sources.

Regardless of the value of the MODE.CHx[PERINTSEL] bit field, software can always force a trigger by using the CONTROL.CHx[PERINTFRC] bit. Likewise, software can always clear a pending DMA trigger using the CONTROL.CHx[PERINTCLR] bit.

Once a particular peripheral trigger event sets a channel's PERINTFLG bit, the bit remains pending until the priority logic of the state machine starts the burst transfer for that channel. Once the burst transfer starts, the flag is cleared. If a new peripheral trigger event is generated while a burst is in progress, the burst will complete before responding to the new peripheral trigger event (after proper prioritization). If a third peripheral trigger event occurs before the pending event is serviced, an error flag is set in the CONTROL.CHx[OVRFLG] bit. If a peripheral trigger event occurs at the same time as the latched flag is being cleared, the trigger event has priority and the PERINTFLG will remain set.

Figure 10-3 shows a diagram of the trigger select circuit. See the DMACHSRCSELx register description for the complete list of peripheral trigger event sources.

Figure 10-3. Peripheral Interrupt Trigger Input Diagram



A See Figure 10-2.

Table 10-1 shows the peripheral trigger source options that are available for each channel.

Table 10-1. DMA Trigger Source Options

Select Value (8-bit)	DMA ChTrigger Source
0	No Peripheral
1	ADCA.1
2	ADCA.2
3	ADCA.3
4	ADCA.4
5	ADCAEVT
6	ADCB.1
7	ADCB.2
8	ADCB.3
9	ADCB.4
10	ADCBEVT
11	ADCC.1
12	ADCC.2
13	ADCC.3
14	ADCC.4
15	ADCCEVT
16 - 28	No Peripheral
29	XINT1

Select Value (8-bit)	DMA ChTrigger Source
30	XINT2
31	XINT3
32	XINT4
33	XINT5
34 - 35	No Peripheral
36	EPWM1.SOCA
37	EPWM1.SOCB
38	EPWM2.SOCA
39	EPWM2.SOCB
40	EPWM3.SOCA
41	EPWM3.SOCB
42	EPWM4.SOCA
43	EPWM4.SOCB
44	EPWM5.SOCA
45	EPWM5.SOCB
46	EPWM6.SOCA
47	EPWM6.SOCB
48	EPWM7.SOCA
49	EPWM7.SOCB
50	EPWM8.SOCA
51	EPWM8.SOCB
52 - 67	No Peripheral
68	TINT0
69	TINT1
70	TINT2
71 - 74	No Peripheral
75	ECAP1DMA
76	ECAP2DMA
77	ECAP3DMA
78	ECAP4DMA
79	ECAP5DMA
80	ECAP6DMA
81	ECAP7DMA
82 - 95	No Peripheral
96	SD1FLT1
97	SD1FLT2
98	SD1FLT3
99	SD1FLT4
100 - 108	No Peripheral
109	SPITXDMAA
110	SPIRXDMAA
111	SPITXDMAB
112	SPIRXDMAB
113 - 116	No Peripheral
117	LINATXDMA
118	LINARXDMA
119 - 122	No Peripheral
123	FSITXADMA
124	No Peripheral



Select Value (8-bit)	DMA ChTrigger Source
125	FSIRXADMA
126	No Peripheral
127	CLB1INT
128	CLB2INT
129	CLB3INT
130	CLB4INT
131 - 166	No Peripheral
167	CANAIF1
168	CANAIF2
169	CANAIF3
170	CANBIF1
171	CANBIF2
172	CANBIF3
173 - 255	No Peripheral

### 10.3.3 DMA Bus

The DMA bus architecture consists of a 32-bit address bus, a 32-bit data read bus, and a 32-bit data write bus. Memories and register locations connected to the DMA bus are via interfaces that sometimes share resources with the CPU memory or peripheral bus. Arbitration rules are defined in [Section 10.6](#).

## 10.4 Address Pointer and Transfer Control

The DMA state machine is, at its most basic level, two nested loops.

### Burst (Inner) Loop:-

The burst (inner) loop transfers programmable number of words set by (BURST\_SIZE + 1) register when a DMA channel trigger (Peripheral / Software trigger) is received. The BURST\_SIZE register allows a maximum of 32 sixteen-bit words to be transferred in one burst. Each DMA channel supports both 16-bit (or) 32-bit word burst which can be controlled by MODE.DATASIZE bit field. Each DMA channel contains a shadowed address pointer for the source (SRC\_ADDR\_SHADOW) and the destination (DST\_ADDR\_SHADOW) address. At the beginning of each transfer, the shadowed version of each pointer is copied into its respective active (SRC\_ADDR\_ACTIVE / DST\_ADDR\_ACTIVE) register. During the burst loop, after each word is transferred, the signed value contained in the appropriate source or destination BURST\_STEP register is added to the active register as shown below

$$\text{SRC\_ADDR\_ACTIVE} = \text{SRC\_ADDR\_ACTIVE} + \text{SRC\_BURST\_STEP}$$

$$\text{DST\_ADDR\_ACTIVE} = \text{DST\_ADDR\_ACTIVE} + \text{DST\_BURST\_STEP}$$

The burst (inner) loop transfers a burst of data when a DMA Channel Trigger (Peripheral / Software trigger) is received.

### Transfer (Outer) Loop:-

The Transfer (outer) loop transfers programmable number of bursts set by (TRANSFER\_SIZE + 1) register for each channel. Since TRANSFER\_SIZE is a 16-bit register, the total size of a transfer allowed is well beyond any practical requirement. During the transfer loop, after each burst is complete, there are two methods that can be used to modify the active address pointer.

**Method 1** (Default): When address wrapping is disabled (SRC\_WRAP\_SIZE / DST\_WRAP\_SIZE is greater than TRANSFER\_SIZE), active address pointer is updated as shown below

$$\text{SRC\_ADDR\_ACTIVE} = \text{SRC\_ADDR\_ACTIVE} + \text{SRC\_TRANSFER\_STEP}$$

$$\text{DST\_ADDR\_ACTIVE} = \text{DST\_ADDR\_ACTIVE} + \text{DST\_TRANSFER\_STEP}$$

**Method 2:** Address wrapping gets enabled when SRC\_WRAP\_SIZE / DST\_WRAP\_SIZE is less than TRANSFER\_SIZE. This allows the channel to wrap multiple times within a single transfer. When the number of bursts is equal to (SRC/DST\_WRAP\_SIZE + 1) register, state machine modifies the active address pointers as shown below.

$$\text{SRC\_BEG\_ADDR\_ACTIVE} = \text{SRC\_BEG\_ADDR\_ACTIVE} + \text{SRC\_WRAP\_STEP}$$

$$\text{DST\_BEG\_ADDR\_ACTIVE} = \text{DST\_BEG\_ADDR\_ACTIVE} + \text{DST\_WRAP\_STEP}$$

$$\text{SRC\_ADDR\_ACTIVE} = \text{SRC\_BEG\_ADDR\_ACTIVE}$$

$$\text{DST\_ADDR\_ACTIVE} = \text{DST\_BEG\_ADDR\_ACTIVE}$$

At the end of DMA transfer, DMA would have transferred (BURST\_SIZE + 1) x (TRANSFER\_SIZE + 1) words.

### OneShot Mode:-

OneShot mode is disabled by default.

When OneShot mode is disabled (MODE.CHx[ONESHOT] = 0), DMA transfers one burst [(BURST\_SIZE + 1) words] of data each time a DMA Channel Trigger is received. After the burst is completed, the state machine moves on to the next pending channel in the priority scheme, even if another trigger for the channel just completed is pending. This feature keeps any single channel from monopolizing the DMA bus.

When OneShot mode is enabled (MODE.CHx[ONESHOT] = 1), DMA transfers all the bursts [(BURST\_SIZE + 1) x (TRANSFER\_SIZE + 1) words] on a single DMA channel trigger. Be careful when using this mode, since this can create a condition where one trigger uses up the majority of the DMA bandwidth.

### Continuos Mode:-

Continuos mode is disabled by default.

When Continuos mode is disabled (MODE.CHx[CONTINUOUS] = 0), DMA state machine disables channel after all bursts in a transfer loop (TRANSFER\_COUNT = 0) are complete. The channel must be re-enabled by setting the RUN bit in the CONTROL register before another transfer can be started on that channel.

When Continuos mode is enabled (MODE.CHx[CONTINUOUS] = 1), DMA state machine keep channel active even after all bursts in a transfer loop (TRANSFER\_COUNT = 0) are complete.

Each DMA channel can trigger its own EPIE interrupt for each DMA transfer either at start of DMA transfer (or) end of DMA transfer using MODE.CHx[CHINTMODE] bit.

**Source/Destination Address Pointers (SRC/DST\_ADDR)**—The value written into the shadow register is the start address of the first location where data is read or written to.

At the beginning of a transfer the shadow register (SRC/DST\_ADDR\_SHADOW) is copied into the active register (SRC/DST\_ADDR\_ACTIVE). The active register performs as the current address pointer.

**Source/Destination Begin Address Pointers (SRC/DST\_BEG\_ADDR)**—This is the wrap pointer.

The value written into the shadow register (SRC/DST\_BEG\_ADDR\_SHADOW) will be loaded into the active register (SRC/DST\_BEG\_ADDR\_ACTIVE) at the start of a transfer. On a wrap condition, the active register (SRC/DST\_BEG\_ADDR\_ACTIVE) will be incremented by the signed value in the appropriate SRC/DST\_WRAP\_STEP register prior to being loaded into the active register ((SRC/DST\_ADDR\_ACTIVE)).

For each channel, the transfer process can be controlled with the following size values:

**Source and Destination Burst Size (BURST\_SIZE):** —This specifies the number of words to be transferred in a burst.

This value is loaded into the BURST\_COUNT register at the beginning of each burst. The BURST\_COUNT decrements each word that is transferred and when it reaches a zero value, the burst is complete, indicating that the next channel can be serviced. The behavior of the current channel is defined by the ONE\_SHOT bit in the MODE register. The maximum size of the burst is dictated by the type of peripheral. For the ADC, the burst size could be all 16 registers (if all 16 registers are used). For RAM the burst size can be up to the maximum allowed by the BURST\_SIZE register, which is 32. See [Table 10-2](#) to understand how BURST\_SIZE register affects the number of 16-bit words transferred with respect to DATASIZE.

**Table 10-2. BURSTSIZE vs DATASIZE Behavior**

BURSTSIZE	Number of 16-bit words transferred in	
	DataSize = 16-bit data	DataSize = 32-bit data
0	1	2
1	2	2
2	3	4
3	4	4
4	5	6
5	6	6
6	7	8
7	8	8
8	9	10
9	10	10
10	11	12
11	12	12
*	*	*
*	*	*
*	*	*
30	31	32
31	32	32

**Source and Destination Transfer Size (TRANSFER\_SIZE):** — This specifies the number of bursts to be transferred per CPU interrupt (if enabled).

Whether this interrupt is generated at the beginning or the end of the transfer is defined in the CHINTMODE bit in the MODE register. Whether the channel remains enabled or not after the transfer is completed is defined by the CONTINUOUS bit in the MODE register. The TRANSFER\_SIZE register is loaded into the TRANSFER\_COUNT register at the beginning of each transfer. The TRANSFER\_COUNT register keeps track of how many bursts of data the channel has transferred and when it reaches zero, the DMA transfer is complete.

**Source/Destination Wrap Size (SRC/DST\_WRAP\_SIZE)**— This specifies the number of bursts to be transferred before the current address pointer wraps around to the beginning.

This feature is used to implement a circular addressing type function. This value is loaded into the appropriate SRC/DST\_WRAP\_COUNT register at the beginning of each transfer. The SRC/DST\_WRAP\_COUNT registers keep track of how many bursts of data the channel has transferred and when they reach zero, the wrap procedure is performed on the appropriate source or destination address pointer. A separate size and count register is allocated for source and destination pointers. To *disable* the wrap function, assign the value of these registers to be larger than the TRANSFER\_SIZE.

---

**NOTE:** The value written to the SIZE registers is one less than the intended size. So, to transfer three 16-bit words, the value 2 should be placed in the SIZE register.

Regardless of the state of the DATASIZE bit, the value specified in the SIZE registers are for 16-bit addresses. So, to transfer three 32-bit words, the value 5 should be placed in the SIZE register.

---

For each source/destination pointer, the address changes can be controlled with the following step values:

**Source/Destination Burst Step (SRC/DST\_BURST\_STEP)**—Within each burst transfer, the address source and destination step sizes are specified by these registers.

This value is a signed 2's complement number so that the address pointer can be incremented or decremented as required. If no increment is desired, such as when accessing the data receive or transmit registers in a communication peripheral, the value of these registers should be set to zero.

**Source/Destination Transfer Step (SRC/DST\_TRANSFER\_STEP)**—This specifies the address offset to start the next burst transfer after completing the current burst transfer.

This is used in cases where registers or data memory locations are spaced at constant intervals. This value is a signed 2's complement number so that the address pointer can be incremented or decremented as required.

**Source/Destination Wrap Step (SRC/DST\_WRAP\_STEP):** —When the wrap counter reaches zero, this value specifies the number of words to add/subtract from the SRC/DST\_BEG\_ADDR pointer and hence sets the new start address.

This implements a circular type of addressing mode, useful in many applications. This value is a signed 2's complement number so that the address pointer can be incremented or decremented as required.

---

**NOTE:** Regardless of the state of the DATASIZE bit, the value specified in the STEP registers are for 16-bit addresses. So, to increment one 32-bit address, a value of 2 should be placed in these registers.

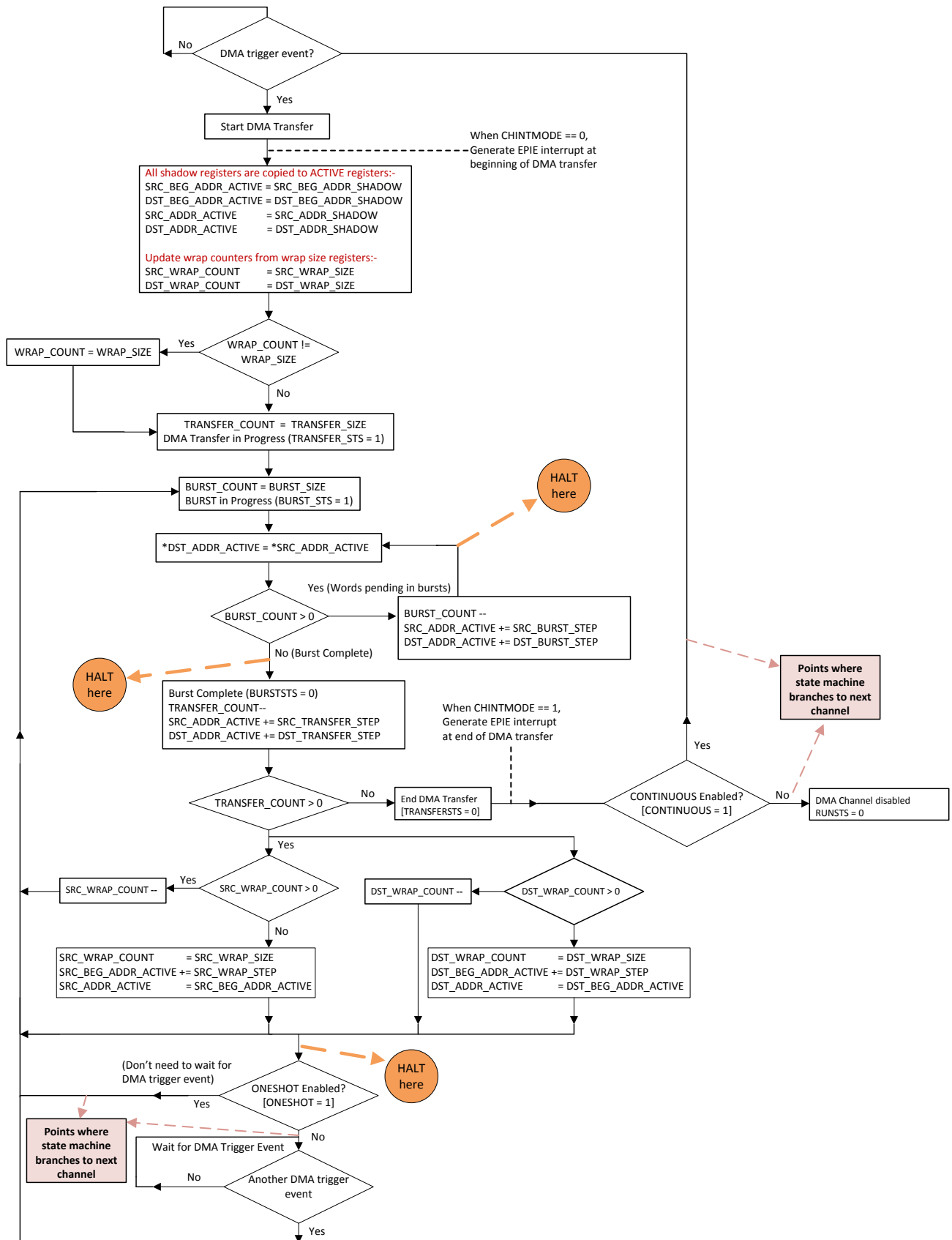
---

**Channel Interrupt Mode (CHINTMODE)**—This mode bit selects whether the DMA interrupt from the respective channel is generated at the beginning of a new transfer or at the end of the transfer.

If implementing a ping-pong buffer scheme with continuous mode of operation, then the interrupt would be generated at the beginning, just after the working registers are copied to the shadow set. If the DMA does not operate in continuous mode, then the interrupt is typically generated at the end when the transfer is complete.

All of the above features and modes are shown in [Figure 10-4](#).

Figure 10-4. DMA State Diagram



The following items are in reference to [Figure 10-4](#).

- The *HALT* points represent where the channel halts operation when interrupted by a high priority channel 1 trigger, or when the HALT command is set, or when an emulation halt is issued and the FREE bit is cleared to 0.
- The SRC/DST\_ADDR\_ACTIVE registers are not affected by SRC/DST\_BEG\_ADDR\_ACTIVE at the start of a transfer. SRC/DST\_BEG\_ADDR\_ACTIVE only affects the SRC/DST\_ADDR\_ACTIVE registers on a wrap. Following is what happens when a transfer first starts:
  - SRC/DST\_BEG\_ADDR\_SHADOW remains unchanged.
  - SRC/DST\_ADDR\_SHADOW remains unchanged.
  - SRC/DST\_BEG\_ADDR\_ACTIVE = SRC/DST\_BEG\_ADDR\_SHADOW
  - SRC/DST\_ADDR\_ACTIVE = SRC/DST\_ADDR\_SHADOW
- The active registers get updated when a wrap occurs. The shadow registers remain unchanged. Specifically:
  - SRC/DST\_BEG\_ADDR\_SHADOW remains unchanged.
  - SRC/DST\_ADDR\_SHADOW remains unchanged.
  - SRC/DST\_BEG\_ADDR\_ACTIVE += SRC/DST\_WRAP\_STEP
  - SRC/DST\_ADDR\_ACTIVE = SRC/DST\_BEG\_ADDR\_ACTIVE
- The best way to remember this is:
  - The shadow registers never change except by software.
  - The active registers never change except by hardware, and a shadow register is only copied into its own active register, never an active register by another name.

## 10.5 Pipeline Timing and Throughput

In addition to the pipeline there are a few other behaviors of the DMA that affect its total throughput:

- A 1-cycle delay is added at the beginning of each burst
- A 1-cycle delay is added when returning from a CH1 high priority interrupt
- Collisions with the CPU may add delay slots (see [Section 10.6](#))
- 32-bit transfers run at double the speed of a 16-bit transfer (it takes the same amount of time to transfer a 32-bit word as it does a 16-bit word)

For example, to transfer 128 16-bit words from GS0 RAM to GS3 RAM, a channel can be configured to transfer 8 bursts of 16 words/burst. This will give:

$$8 \text{ bursts} * [(3 \text{ cycles/word} * 16 \text{ words/burst}) + 1] = 392 \text{ cycles}$$

If instead the channel were configured to transfer the same amount of data 32 bits at a time (the word size is configured to 32 bits) the transfer would take:

$$8 \text{ bursts} * [(3 \text{ cycles/word} * 8 \text{ words/burst}) + 1] = 200 \text{ cycles}$$

The DMA module consists of a 3-stage pipeline as shown in Figure 10-5 and Figure 10-6.

Figure 10-5. 3-Stage Pipeline DMA Transfer

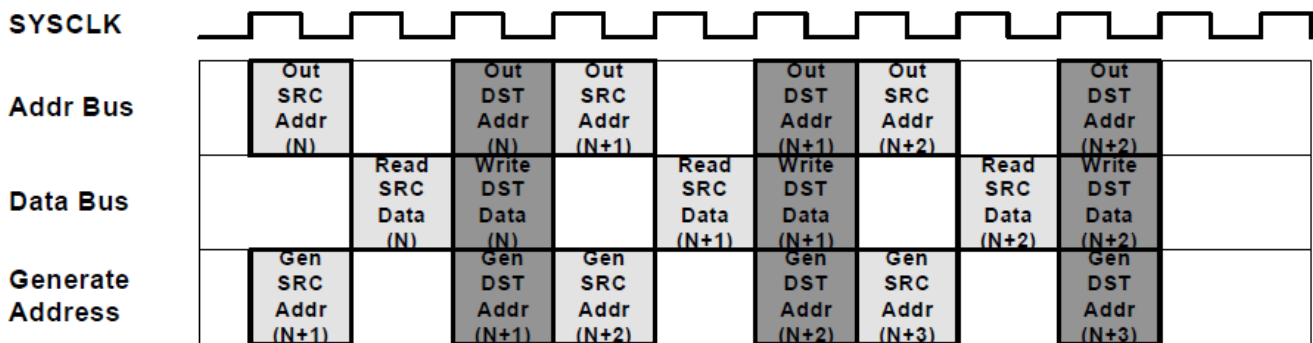
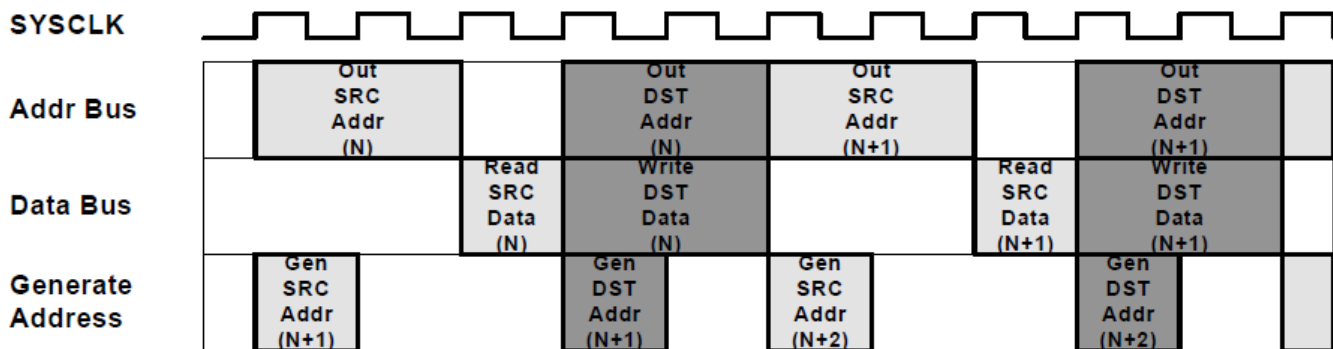


Figure 10-6. 3-stage Pipeline With One Read Stall



## 10.6 CPU and CLA Arbitration

Typically, DMA activity is independent of CPU and CLA activity. However, when the DMA and CPU (or CLA) try to access the same peripheral at the same time, an arbitration procedure is required to resolve the conflict. All instances of the same peripheral type conflict with each other. For instance, CAN-A and CAN-B conflict, as do the GS0 and GS2 RAMs. Different peripheral types can share a bus interface, which creates further opportunities for conflicts. These bus interfaces are:

- Peripheral frame 1: ePWM, eCAP, eQEP, SDFM, CMPSS, DAC
- Peripheral frame 2: PMBus and SPI

**Conflict Example:** The CLA is accessing DAC-A while the DMA is simultaneously accessing DAC-B.

**Conflict Example:** The CPU is accessing an SPI FIFO while the DMA is simultaneously accessing a PMBus register.

**Non-conflict Example:** The CPU is accessing a shared ePWM while the DMA is accessing an SPI.

The exception to all this is the ADC result registers, which are duplicated for each bus master. The CPU, DMA, and CLA can all simultaneously read these registers with no stalls for any master.

A DMA transfer consists of four phases: send source address, read source data, send destination address, and write destination data (see Section 10.5). Suppose CPU accesses a peripheral / memory causing conflict in middle of a DMA transfer, CPU is stalled till the current DMA access is complete and not until the completion of whole DMA transfer.

The following priority schemes are implemented for the various interfaces on the device.

- The fixed priority scheme for the peripheral frames is:
  - CLA/DMA Write
  - CLA/DMA Read



- CPU Write
- CPU Read
- The priority scheme for GSx RAM accesses is round-robin.
- All masters can access the ADC result registers simultaneously without delay.

---

**NOTE:** If the CPU is performing a read-modify-write operation and the DMA performs a write to the same location, the DMA write may be lost if the operation occurs in between the CPU read and the CPU write. Avoid mixing CPU writes with DMA writes to the same locations.

---

Arbitration within DMA channels is based on a round robin priority (or) Channel 1 high priority scheme described in [Section 10.7](#).

## 10.7 Channel Priority

Two priority schemes exist when determining channel priority: Round-robin mode and Channel 1 high-priority mode.

### 10.7.1 Round-Robin Mode

In this mode, all channels have *equal* priority and each enabled channel is serviced in round-robin fashion as follows:

$$\text{CH1} \rightarrow \text{CH2} \rightarrow \text{CH3} \rightarrow \text{CH4} \rightarrow \text{CH5} \rightarrow \text{CH6} \rightarrow \text{CH1} \rightarrow \text{CH2} \rightarrow \dots$$

In the case above, after each channel has transferred a burst of words, the next channel is serviced. You can specify the size of the burst for each channel. Once CH6 (or the last enabled channel) has been serviced, and no other channels are pending, the round-robin state machine enters an idle state.

From the idle state, channel 1 (if enabled) is always serviced first. However, if the DMA is currently processing another channel *x*, all other pending channels between *x* and the end of the round are serviced before CH1. It is in this sense that all the channels are of *equal* priority. For instance, take an example where CH1, CH4, and CH5 are enabled in round-robin mode and CH4 is currently being processed. Then CH1 and CH5 both receive an interrupt trigger from their respective peripherals before CH4 completes. CH1 and CH5 are now both pending. When CH4 completes its burst, CH5 will be serviced next. Only after CH5 completes will CH1 be serviced. Upon completion of CH1, if there are no more channels pending, the round-robin state machine will enter an idle state.

A more complicated example is shown below:

- Assume all channels are enabled, and the DMA is in an idle state,
- Initially a trigger occurs on CH1, CH3, and CH5 on the same cycle,
- When the CH1 burst transfer starts, requests from CH3 and CH5 are pending,
- Before completion of the CH1 burst, the DMA receives a request from CH2. Now the pending requests are from CH2, CH3, and CH5,
- After completing the CH1 burst, CH2 will be serviced since it is next in the round-robin scheme after CH1.
- After the burst from CH2 is finished, the CH3 burst will be serviced, followed by CH5 burst.
- Now while the CH5 burst is being serviced, the DMA receives a request from CH1, CH3, and CH6.
- The burst from CH6 will start after the completion of the CH5 burst since it is the next channel after CH5 in the round-robin scheme.
- This will be followed by the CH1 burst and then the CH3 burst
- After the CH3 burst finishes, assuming no more triggers have occurred, the round-robin state machine will enter an idle state.

The round-robin state machine may be reset to the idle state via the DMACTRL[PRIORITYRESET] bit.



### 10.7.2 Channel 1 High Priority Mode

In this mode, Channel 1 has high priority over all the other channels. Channel 2 – 6 have equal priority and each enabled channel is serviced in round-robin fashion.

Higher Priority: CH1  
 Lower priority: CH2 → CH3 → CH4 → CH5 → CH6 → CH2 → ...

Given an example where CH1, CH4 and CH5 are enabled in Channel 1 High Priority Mode and CH4 is currently being processed. Then CH1 and CH5 both receive an interrupt trigger from their respective peripherals before CH4 completes. CH1 and CH5 are now both pending. When the current CH4 word transfer is completed, regardless of whether the DMA has completed the entire CH4 burst, CH4 execution will be suspended and CH1 will be serviced. After the CH1 burst completes, CH4 will resume execution.

Upon completion of CH4, CH5 will be serviced. After CH5 completes, if there are no more channels pending, the round-robin state machine will enter an idle state.

Typically Channel 1 would be used in this mode for the ADC, since its data rate is so high. However, Channel 1 High Priority Mode may be used in conjunction with any peripheral.

---

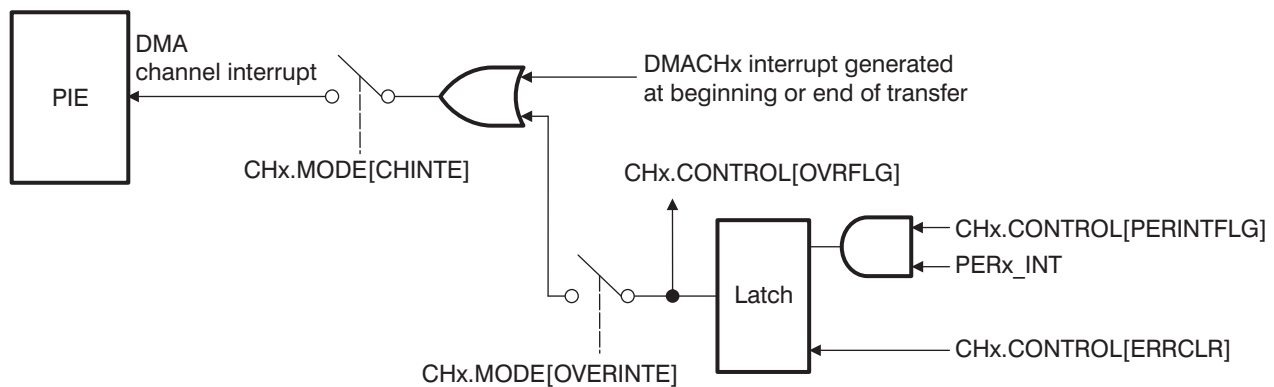
**NOTE:** High-priority mode and ONESHOT mode may not be used at the same time on channel 1. Other channels may use ONESHOT mode when channel 1 is in high-priority mode.

---

### 10.8 Overrun Detection Feature

The DMA contains overrun detection logic. When a peripheral event trigger is received by the DMA, the PERINTFLG bit in the CONTROL register is set, pending the channel to the DMA state machine. When the burst for that channel is started, the PERINTFLG is cleared. If however, between the time that the PERINTFLG bit is set by an event trigger and cleared by the start of the burst, an additional event trigger arrives, the second trigger will be lost. This condition will set the OVRFLG bit in the CONTROL register as in Figure 10-7. If the overrun interrupt is enabled, the channel interrupt will be generated to the PIE module.

Figure 10-7. Overrun Detection Logic



## 10.9 DMA Registers

This section describes the C28x Direct Memory Access Registers.

### 10.9.1 DMA Base Addresses

**Table 10-3. DMA Base Address Table**

Device Registers	Register Name	Start Address	End Address
DmaRegs	DMA_REGS	0x0000_1000	0x0000_11FF

### 10.9.2 DMA\_REGS Registers

Table 10-4 lists the DMA\_REGS registers. All register offset addresses not listed in Table 10-4 should be considered as reserved locations and the register contents should not be modified.

**Table 10-4. DMA\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	DMACTRL	DMA Control Register	EALLOW	<a href="#">Go</a>
1h	DEBUGCTRL	Debug Control Register	EALLOW	<a href="#">Go</a>
4h	PRIORITYCTRL1	Priority Control 1 Register	EALLOW	<a href="#">Go</a>
6h	PRIORITYSTAT	Priority Status Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 10-5 shows the codes that are used for access types in this section.

**Table 10-5. DMA\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 10.9.2.1 DMACTRL Register (Offset = 0h) [reset = 0h]

DMACTRL is shown in [Figure 10-8](#) and described in [Table 10-6](#).

Return to the [Summary Table](#).

DMA Control Register

**Figure 10-8. DMACTRL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						PRIORITYRES ET	HARDRESET
R-0h						R-0/W1S-0h	R-0/W1S-0h

**Table 10-6. DMACTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1	PRIORITYRESET	R-0/W1S	0h	<p>The priority reset bit resets the round-robin state machine when a 1 is written. Service starts from the first enabled channel. Writes of 0 are ignored and this bit always reads back a 0.</p> <p>When a 1 is written to this bit, any pending burst transfer completes before resetting the channel priority machine. If CH1 is configured as a high-priority channel, and this bit is written to while CH1 is servicing a burst, both the CH1 burst and the next pending low-priority burst are completed before the state machine is reset.</p> <p>If CH1 is high-priority, the state machine restarts from CH2 (or the next highest enabled channel).</p> <p>Reset type: SYSRSn</p>
0	HARDRESET	R-0/W1S	0h	<p>Writing a 1 to the hard reset bit resets the whole DMA and aborts any current access (similar to applying a device reset). Writes of 0 are ignored and this bit always reads back a 0.</p> <p>For a soft reset, a bit is provided for each channel to perform a gentler reset. Refer to the channel control registers.</p> <p>When writing to this bit, there is a one cycle delay before it takes effect. Hence, a one-cycle delay (such as a NOP instruction) is required in software before attempting to access any other DMA register.</p> <p>Reset type: SYSRSn</p>

### 10.9.2.2 DEBUGCTRL Register (Offset = 1h) [reset = 0h]

DEBUGCTRL is shown in [Figure 10-9](#) and described in [Table 10-7](#).

Return to the [Summary Table](#).

Debug Control Register

**Figure 10-9. DEBUGCTRL Register**

15	14	13	12	11	10	9	8
FREE	RESERVED						
R/W-0h	R-0h						
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

**Table 10-7. DEBUGCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	FREE	R/W	0h	Emulation Control  This bit specifies the action when an emulation halt event occurs. Reset type: SYSRSn 0h (R/W) = The DMA completes the current read-write operation, then halts. 1h (R/W) = The DMA continues running during an emulation halt.
14-0	RESERVED	R	0h	Reserved

### 10.9.2.3 PRIORITYCTRL1 Register (Offset = 4h) [reset = 0h]

PRIORITYCTRL1 is shown in [Figure 10-10](#) and described in [Table 10-8](#).

Return to the [Summary Table](#).

Priority Control 1 Register

**Figure 10-10. PRIORITYCTRL1 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							CH1PRIORITY
R-0h							R/W-0h

**Table 10-8. PRIORITYCTRL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	CH1PRIORITY	R/W	0h	DMA Channel 1 Priority  This bit selects whether CH1 has high priority or not. The priority can only be changed when all channels are disabled. A priority reset should be performed before restarting channels after changing priority  Reset type: SYSRSn 0h (R/W) = CH1 has the same priority as the other channels 1h (R/W) = CH1 has a higher priority than the other channels

### 10.9.2.4 PRIORITYSTAT Register (Offset = 6h) [reset = 0h]

PRIORITYSTAT is shown in [Figure 10-11](#) and described in [Table 10-9](#).

Return to the [Summary Table](#).

Priority Status Register

**Figure 10-11. PRIORITYSTAT Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	ACTIVESTS_SHADOW			RESERVED	ACTIVESTS		
R-0h	R-0h			R-0h	R-0h		

**Table 10-9. PRIORITYSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6-4	ACTIVESTS_SHADOW	R	0h	Active Channel Status Shadow  These bits are only meaningful when CH1 is in high-priority mode. When CH1 is serviced, the ACTIVESTS bits are copied to the shadow bits and indicate which channel was interrupted by CH1. When CH1 service is completed, the shadow bits are copied back to the ACTIVESTS bits. If this bit field is zero or the same as the ACTIVESTS bit field, then no channel is pending due to a CH1 interrupt. When CH1 is not a higher priority channel, these bits should be ignored.  Reset type: SYSRSn 0h (R/W) = No channel is active 1h (R/W) = CH 1 2h (R/W) = CH 2 3h (R/W) = CH 3 4h (R/W) = CH 4 5h (R/W) = CH 5 6h (R/W) = CH 6 7h (R/W) = Reserved
3	RESERVED	R	0h	Reserved
2-0	ACTIVESTS	R	0h	Active Channel Status  These bits indicate which channel (if any) is currently active or performing a transfer.  Reset type: SYSRSn 0h (R/W) = No channel is active 1h (R/W) = CH 1 2h (R/W) = CH 2 3h (R/W) = CH 3 4h (R/W) = CH 4 5h (R/W) = CH 5 6h (R/W) = CH 6 7h (R/W) = Reserved

### 10.9.3 DMA\_CH\_REGS Registers

Table 10-10 lists the DMA\_CH\_REGS registers. All register offset addresses not listed in Table 10-10 should be considered as reserved locations and the register contents should not be modified.

**Table 10-10. DMA\_CH\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	MODE	Mode Register	EALLOW	<a href="#">Go</a>
1h	CONTROL	Control Register	EALLOW	<a href="#">Go</a>
2h	BURST_SIZE	Burst Size Register	EALLOW	<a href="#">Go</a>
3h	BURST_COUNT	Burst Count Register	EALLOW	<a href="#">Go</a>
4h	SRC_BURST_STEP	Source Burst Step Register	EALLOW	<a href="#">Go</a>
5h	DST_BURST_STEP	Destination Burst Step Register	EALLOW	<a href="#">Go</a>
6h	TRANSFER_SIZE	Transfer Size Register	EALLOW	<a href="#">Go</a>
7h	TRANSFER_COUNT	Transfer Count Register	EALLOW	<a href="#">Go</a>
8h	SRC_TRANSFER_STEP	Source Transfer Step Register	EALLOW	<a href="#">Go</a>
9h	DST_TRANSFER_STEP	Destination Transfer Step Register	EALLOW	<a href="#">Go</a>
Ah	SRC_WRAP_SIZE	Source Wrap Size Register	EALLOW	<a href="#">Go</a>
Bh	SRC_WRAP_COUNT	Source Wrap Count Register	EALLOW	<a href="#">Go</a>
Ch	SRC_WRAP_STEP	Source Wrap Step Register	EALLOW	<a href="#">Go</a>
Dh	DST_WRAP_SIZE	Destination Wrap Size Register	EALLOW	<a href="#">Go</a>
Eh	DST_WRAP_COUNT	Destination Wrap Count Register	EALLOW	<a href="#">Go</a>
Fh	DST_WRAP_STEP	Destination Wrap Step Register	EALLOW	<a href="#">Go</a>
10h	SRC_BEG_ADDR_SHADOW	Source Begin Address Shadow Register	EALLOW	<a href="#">Go</a>
12h	SRC_ADDR_SHADOW	Source Address Shadow Register	EALLOW	<a href="#">Go</a>
14h	SRC_BEG_ADDR_ACTIVE	Source Begin Address Active Register	EALLOW	<a href="#">Go</a>
16h	SRC_ADDR_ACTIVE	Source Address Active Register	EALLOW	<a href="#">Go</a>
18h	DST_BEG_ADDR_SHADOW	Destination Begin Address Shadow Register	EALLOW	<a href="#">Go</a>
1Ah	DST_ADDR_SHADOW	Destination Address Shadow Register	EALLOW	<a href="#">Go</a>
1Ch	DST_BEG_ADDR_ACTIVE	Destination Begin Address Active Register	EALLOW	<a href="#">Go</a>
1Eh	DST_ADDR_ACTIVE	Destination Address Active Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 10-11 shows the codes that are used for access types in this section.

**Table 10-11. DMA\_CH\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R -0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W 1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		



**Table 10-11. DMA\_CH\_REGS Access Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 10.9.3.1 MODE Register (Offset = 0h) [reset = 0h]

MODE is shown in [Figure 10-12](#) and described in [Table 10-12](#).

Return to the [Summary Table](#).

Mode Register

**Figure 10-12. MODE Register**

15		14		13		12		11		10		9		8	
CHINTE	DATASIZE	RESERVED	RESERVED	RESERVED	RESERVED	CONTINUOUS	ONESHOT	CHINTMODE	PERINTE						
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	
OVRINTE		RESERVED								PERINTSEL					
R/W-0h			R-0h								R/W-0h				

**Table 10-12. MODE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	CHINTE	R/W	0h	Channel Interrupt Enable Bit  This bit enables the DMA channel's CPU interrupt. Reset type: SYSRSn 0h (R/W) = Interrupt disabled 1h (R/W) = Interrupt enabled
14	DATASIZE	R/W	0h	Data Size Mode Bit  This bit determines whether the DMA channel transfers 16 bits or 32 bits of data per read/write operation. Regardless of this setting, all data lengths and offsets in other DMA registers refer to 16-bit words. The pointer step increments must be configured to accommodate 32-bit words. Reset type: SYSRSn 0h (R/W) = 16-bit data transfer size 1h (R/W) = 32-bit data transfer size
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	CONTINUOUS	R/W	0h	Continuous Mode Bit  If this bit is set to 1, then the channel re-initializes when TRANSFER_COUNT is zero and waits for the next event trigger. Otherwise, the DMA stops and clears the RUNSTS bit. Reset type: SYSRSn
10	ONESHOT	R/W	0h	One Shot Mode  If this bit is set to 1, each peripheral event trigger causes the channel to perform an entire transfer. Otherwise, the channel only performs one burst per trigger. Reset type: SYSRSn
9	CHINTMODE	R/W	0h	Channel Interrupt Generation Mode  This bit specifies when the DMA channel generates a CPU interrupt for a transfer. Reset type: SYSRSn 0h (R/W) = Generate interrupt at beginning of new transfer 1h (R/W) = Generate interrupt at end of transfer.

**Table 10-12. MODE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	PERINTE	R/W	0h	Peripheral Event Trigger Enable  This bit enables peripheral event triggers on the DMA channel. Reset type: SYSRSn 0h (R/W) = Peripheral event trigger disabled. Neither the selected peripheral nor software can start a DMA burst. 1h (R/W) = Peripheral event trigger enabled.
7	OVRINTE	R/W	0h	Overflow Interrupt Enable  The bit determines whether the DMA module generates a CPU interrupt when it detects an overflow event. Reset type: SYSRSn 0h (R/W) = Overflow interrupt disabled 1h (R/W) = Overflow interrupt enabled
6-5	RESERVED	R	0h	Reserved
4-0	PERINTSEL	R/W	0h	Peripheral Event Trigger Source Select  These are legacy bits and should be set to the channel number. The actual source selection is done via the DMACHSRCSELn registers, which are part of the DMA_CLA_SRC_SEL_REGS group. Reset type: SYSRSn

### 10.9.3.2 CONTROL Register (Offset = 1h) [reset = 0h]

CONTROL is shown in [Figure 10-13](#) and described in [Table 10-13](#).

Return to the [Summary Table](#).

Control Register

**Figure 10-13. CONTROL Register**

15	14	13	12	11	10	9	8
RESERVED	OVRFLG	RUNSTS	BURSTSTS	TRANSFERSTS	RESERVED	RESERVED	PERINTFLG
R-0h	R-0h	R-0h	R-0h	R-0h			R-0h
7	6	5	4	3	2	1	0
ERRCLR	RESERVED	RESERVED	PERINTCLR	PERINTFRC	SOFTRESET	HALT	RUN
R-0/W1S-0h			R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 10-13. CONTROL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	OVRFLG	R	0h	Overflow Flag  This bit indicates that a peripheral event trigger was received while PERINTFLG was already set. It can be cleared by writing to the ERRCLR bit.  Reset type: SYSRSn 0h (R/W) = No overflow detected 1h (R/W) = Overflow detected
13	RUNSTS	R	0h	Run Status Flag  This bit indicates that the DMA channel is ready to respond to peripheral event triggers. This bit is set when a 1 is written to the RUN bit. It is cleared when a transfer completes (TRANSFER_COUNT = 0) and continuous mode is disabled, or when the HARDRESET, SOFTRESET, or HALT bit is set.  Reset type: SYSRSn 0h (R/W) = The channel is disabled 1h (R/W) = The channel is enabled
12	BURSTSTS	R	0h	Burst Status Flag  This bit is set when a DMA burst begins. The BURST_COUNT is set to the BURST_SIZE. This bit is cleared when BURST_COUNT reaches zero, or when the HARDRESET or SOFTRESET bit is set.  Reset type: SYSRSn 0h (R/W) = No burst activity 1h (R/W) = The DMA is currently servicing or suspending a burst transfer from this channel
11	TRANSFERSTS	R	0h	Transfer Status Flag  This bit is set when a DMA transfer begins. The address registers are copied to the shadow set and the TRANSFER_COUNT is set to the TRANSFER_SIZE. This bit is cleared when TRANSFER_COUNT reaches zero, or when the HARDRESET or SOFTRESET bit is set.  Reset type: SYSRSn 0h (R/W) = No transfer activity 1h (R/W) = The channel is currently in the middle of a transfer regardless of whether a burst of data is actively being transferred or not
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved

**Table 10-13. CONTROL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	PERINTFLG	R	0h	Peripheral Event Trigger Flag  This bit indicates whether a peripheral event trigger has arrived. This bit is automatically cleared when the first burst transfer begins. Reset type: SYSRSn 0h (R/W) = Waiting for event trigger 1h (R/W) = Event trigger pending
7	ERRCLR	R-0/W1S	0h	Clear Error  Writing a 1 to this bit will clear the OVRFLG bit. This is normally done when initializing the DMA module or if an overflow condition is detected. If an overflow event occurs at the same time this bit is set, the overrun has priority and the OVRFLG bit is set. Reset type: SYSRSn
6	RESERVED	R-0/W1S	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	PERINTCLR	R-0/W1S	0h	Clear Peripheral Event Trigger  Writing a 1 to this bit clears PERINTFLG, which cancels a pending event trigger. This is normally done when initializing the DMA module. If an event trigger arrives at the same time this bit is set, the trigger has priority and PERINTFLG is set. Reset type: SYSRSn
3	PERINTFRC	R-0/W1S	0h	Force Peripheral Event Trigger  If the PERINTE bit of the MODE register is set, writing a 1 to this bit sets PERINTFLG, which triggers a DMA burst. This bit can be used to start a DMA transfer in software. Reset type: SYSRSn
2	SOFTRESET	R-0/W1S	0h	Channel Soft Reset  Writing a 1 to this bit places the channel into its default state after the current read/write access has completed:  RUNSTS = 0 TRANSFERSTS = 0 BURSTSTS = 0 BURST_COUNT = 0 TRANSFER_COUNT = 0 SRC_WRAP_COUNT = 0 DST_WRAP_COUNT = 0 Reset type: SYSRSn
1	HALT	R-0/W1S	0h	Halt Channel  Writing a 1 to this bit halts the DMA channel in its current state after any ongoing read/write access has completed. Reset type: SYSRSn
0	RUN	R-0/W1S	0h	Run Channel  Writing a 1 to this bit enables the DMA channel and sets the RUNSTS bit to 1. This bit is also used to resume after a channel halt.  The RUN bit is typically used to start the DMA channel after configuration. The channel will then wait for the first peripheral event trigger (PERINTFLG == 1) to start a burst. Reset type: SYSRSn

### 10.9.3.3 BURST\_SIZE Register (Offset = 2h) [reset = 0h]

BURST\_SIZE is shown in [Figure 10-14](#) and described in [Table 10-14](#).

Return to the [Summary Table](#).

Burst Size Register

**Figure 10-14. BURST\_SIZE Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				BURSTSIZE			
R-0h				R/W-0h			

**Table 10-14. BURST\_SIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4-0	BURSTSIZE	R/W	0h	These bits specify the burst size in 16-bit words. The actual size is equal to BURSTSIZE + 1. Reset type: SYSRSn

**10.9.3.4 BURST\_COUNT Register (Offset = 3h) [reset = 0h]**

 BURST\_COUNT is shown in [Figure 10-15](#) and described in [Table 10-15](#).

 Return to the [Summary Table](#).

Burst Count Register

**Figure 10-15. BURST\_COUNT Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				BURSTCOUNT			
R-0h				R-0h			

**Table 10-15. BURST\_COUNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4-0	BURSTCOUNT	R	0h	These bits indicate the number of words left in the current burst. Reset type: SYSRSn 0h (R/W) = 0 word left in a burst 1h (R/W) = 1 word left in a burst 2h (R/W) = 2 word left in a burst 3h (R/W) = 3 word left in a burst 4h (R/W) = 4 word left in a burst 5h (R/W) = 5 word left in a burst 6h (R/W) = 6 word left in a burst 7h (R/W) = 7 word left in a burst 8h (R/W) = 8 word left in a burst 9h (R/W) = 9 word left in a burst Ah (R/W) = 10 word left in a burst Bh (R/W) = 11 word left in a burst Ch (R/W) = 12 word left in a burst Dh (R/W) = 13 word left in a burst Eh (R/W) = 14 word left in a burst Fh (R/W) = 15 word left in a burst 10h (R/W) = 16 word left in a burst 11h (R/W) = 17 word left in a burst 12h (R/W) = 18 word left in a burst 13h (R/W) = 19 word left in a burst 14h (R/W) = 20 word left in a burst 15h (R/W) = 21 word left in a burst 16h (R/W) = 22 word left in a burst 17h (R/W) = 23 word left in a burst 18h (R/W) = 24 word left in a burst 19h (R/W) = 25 word left in a burst 1Ah (R/W) = 26 word left in a burst 1Bh (R/W) = 27 word left in a burst 1Ch (R/W) = 28 word left in a burst 1Dh (R/W) = 29 word left in a burst 1Eh (R/W) = 30 word left in a burst 1Fh (R/W) = 31 word left in a burst

### 10.9.3.5 SRC\_BURST\_STEP Register (Offset = 4h) [reset = 0h]

SRC\_BURST\_STEP is shown in [Figure 10-16](#) and described in [Table 10-16](#).

Return to the [Summary Table](#).

Source Burst Step Register

**Figure 10-16. SRC\_BURST\_STEP Register**

15	14	13	12	11	10	9	8
SRCBURSTSTEP							
R/W-0h							
7	6	5	4	3	2	1	0
SRCBURSTSTEP							
R/W-0h							

**Table 10-16. SRC\_BURST\_STEP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SRCBURSTSTEP	R/W	0h	<p>These bits specify the change in the source address after each word in a burst. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the source address after each read/write operation in a burst.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No address change            1h (R/W) = Add 1 to the address            2h (R/W) = Add 2 to the address            FFEh (R/W) = Add 4094 to the address            FFFh (R/W) = Add 4095 to the address            F00h (R/W) = Subtract 4096 from the address            F01h (R/W) = Subtract 4095 from the address            FFFEh (R/W) = Subtract 2 from the address            FFFFh (R/W) = Subtract 1 from the address</p>



### 10.9.3.6 DST\_BURST\_STEP Register (Offset = 5h) [reset = 0h]

DST\_BURST\_STEP is shown in [Figure 10-17](#) and described in [Table 10-17](#).

Return to the [Summary Table](#).

Destination Burst Step Register

**Figure 10-17. DST\_BURST\_STEP Register**

15	14	13	12	11	10	9	8
DSTBURSTSTEP							
R/W-0h							
7	6	5	4	3	2	1	0
DSTBURSTSTEP							
R/W-0h							

**Table 10-17. DST\_BURST\_STEP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DSTBURSTSTEP	R/W	0h	<p>These bits specify the change in the destination address after each word in a burst. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the destination address after each read/write operation in a burst.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No address change            1h (R/W) = Add 1 to the address            2h (R/W) = Add 2 to the address            FFEh (R/W) = Add 4094 to the address            FFFh (R/W) = Add 4095 to the address            F00h (R/W) = Subtract 4096 from the address            F01h (R/W) = Subtract 4095 from the address            FFFEh (R/W) = Subtract 2 from the address            FFFFh (R/W) = Subtract 1 from the address</p>

### 10.9.3.7 TRANSFER\_SIZE Register (Offset = 6h) [reset = 0h]

TRANSFER\_SIZE is shown in [Figure 10-18](#) and described in [Table 10-18](#).

Return to the [Summary Table](#).

Transfer Size Register

**Figure 10-18. TRANSFER\_SIZE Register**

15	14	13	12	11	10	9	8
TRANSFERSIZE							
R/W-0h							
7	6	5	4	3	2	1	0
TRANSFERSIZE							
R/W-0h							

**Table 10-18. TRANSFER\_SIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TRANSFERSIZE	R/W	0h	These bits specify the transfer size in bursts. The actual size is equal to TRANSFERSIZE + 1. Reset type: SYSRSn

### 10.9.3.8 TRANSFER\_COUNT Register (Offset = 7h) [reset = 0h]

TRANSFER\_COUNT is shown in [Figure 10-19](#) and described in [Table 10-19](#).

Return to the [Summary Table](#).

Transfer Count Register

**Figure 10-19. TRANSFER\_COUNT Register**

15	14	13	12	11	10	9	8
TRANSFERCOUNT							
R/W-0h							
7	6	5	4	3	2	1	0
TRANSFERCOUNT							
R/W-0h							

**Table 10-19. TRANSFER\_COUNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TRANSFERCOUNT	R/W	0h	These bits indicate the number of bursts left in the current transfer. Reset type: SYSRSn

### 10.9.3.9 SRC\_TRANSFER\_STEP Register (Offset = 8h) [reset = 0h]

SRC\_TRANSFER\_STEP is shown in [Figure 10-20](#) and described in [Table 10-20](#).

Return to the [Summary Table](#).

Source Transfer Step Register

**Figure 10-20. SRC\_TRANSFER\_STEP Register**

15	14	13	12	11	10	9	8
SRCTRANSFERSTEP							
R/W-0h							
7	6	5	4	3	2	1	0
SRCTRANSFERSTEP							
R/W-0h							

**Table 10-20. SRC\_TRANSFER\_STEP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SRCTRANSFERSTEP	R/W	0h	<p>These bits specify the change in the source address after a burst completes. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the source address after each burst completes.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No address change</p> <p>1h (R/W) = Add 1 to the address</p> <p>2h (R/W) = Add 2 to the address</p> <p>FFEh (R/W) = Add 4094 to the address</p> <p>FFFh (R/W) = Add 4095 to the address</p> <p>F000h (R/W) = Subtract 4096 from the address</p> <p>F001h (R/W) = Subtract 4095 from the address</p> <p>FFFEh (R/W) = Subtract 2 from the address</p> <p>FFFFh (R/W) = Subtract 1 from the address</p>

**10.9.3.10 DST\_TRANSFER\_STEP Register (Offset = 9h) [reset = 0h]**

DST\_TRANSFER\_STEP is shown in [Figure 10-21](#) and described in [Table 10-21](#).

Return to the [Summary Table](#).

Destination Transfer Step Register

**Figure 10-21. DST\_TRANSFER\_STEP Register**

15	14	13	12	11	10	9	8
DSTTRANSFERSTEP							
R/W-0h							
7	6	5	4	3	2	1	0
DSTTRANSFERSTEP							
R/W-0h							

**Table 10-21. DST\_TRANSFER\_STEP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DSTTRANSFERSTEP	R/W	0h	<p>These bits specify the change in the destination address after a burst completes. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the destination address after each burst completes.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No address change</p> <p>1h (R/W) = Add 1 to the address</p> <p>2h (R/W) = Add 2 to the address</p> <p>FFeh (R/W) = Add 4094 to the address</p> <p>FFFh (R/W) = Add 4095 to the address</p> <p>F00h (R/W) = Subtract 4096 from the address</p> <p>F01h (R/W) = Subtract 4095 from the address</p> <p>FFFEh (R/W) = Subtract 2 from the address</p> <p>FFFFh (R/W) = Subtract 1 from the address</p>

**10.9.3.11 SRC\_WRAP\_SIZE Register (Offset = Ah) [reset = 0h]**

SRC\_WRAP\_SIZE is shown in [Figure 10-22](#) and described in [Table 10-22](#).

Return to the [Summary Table](#).

Source Wrap Size Register

**Figure 10-22. SRC\_WRAP\_SIZE Register**

15	14	13	12	11	10	9	8
WRAPSIZE							
R/W-0h							
7	6	5	4	3	2	1	0
WRAPSIZE							
R/W-0h							

**Table 10-22. SRC\_WRAP\_SIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	WRAPSIZE	R/W	0h	These bits specify the number of bursts to transfer before the source address wraps around to the beginning address. The actual number is equal to WRAPSIZE + 1. To disable the wrapping function, set WRAPSIZE to a value larger than TRANSFERSIZE. Reset type: SYSRSn

**10.9.3.12 SRC\_WRAP\_COUNT Register (Offset = Bh) [reset = 0h]**

SRC\_WRAP\_COUNT is shown in [Figure 10-23](#) and described in [Table 10-23](#).

Return to the [Summary Table](#).

Source Wrap Count Register

**Figure 10-23. SRC\_WRAP\_COUNT Register**

15	14	13	12	11	10	9	8
WRAPSIZE							
R/W-0h							
7	6	5	4	3	2	1	0
WRAPSIZE							
R/W-0h							

**Table 10-23. SRC\_WRAP\_COUNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	WRAPSIZE	R/W	0h	These bits indicate the number of bursts left before wrapping the source address. Reset type: SYSRSn

### 10.9.3.13 SRC\_WRAP\_STEP Register (Offset = Ch) [reset = 0h]

SRC\_WRAP\_STEP is shown in [Figure 10-24](#) and described in [Table 10-24](#).

Return to the [Summary Table](#).

Source Wrap Step Register

**Figure 10-24. SRC\_WRAP\_STEP Register**

15	14	13	12	11	10	9	8
WRAPSTEP							
R/W-0h							
7	6	5	4	3	2	1	0
WRAPSTEP							
R/W-0h							

**Table 10-24. SRC\_WRAP\_STEP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	WRAPSTEP	R/W	0h	<p>These bits specify the change in the source beginning address when the wrap counter reaches zero. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the source address when wrapping occurs.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No address change            1h (R/W) = Add 1 to the address            2h (R/W) = Add 2 to the address            FFEh (R/W) = Add 4094 to the address            FFFh (R/W) = Add 4095 to the address            F00h (R/W) = Subtract 4096 from the address            F001h (R/W) = Subtract 4095 from the address            FFFEh (R/W) = Subtract 2 from the address            FFFFh (R/W) = Subtract 1 from the address</p>



**10.9.3.14 DST\_WRAP\_SIZE Register (Offset = Dh) [reset = 0h]**

DST\_WRAP\_SIZE is shown in [Figure 10-25](#) and described in [Table 10-25](#).

Return to the [Summary Table](#).

Destination Wrap Size Register

**Figure 10-25. DST\_WRAP\_SIZE Register**

15	14	13	12	11	10	9	8
WRAPSIZE							
R/W-0h							
7	6	5	4	3	2	1	0
WRAPSIZE							
R/W-0h							

**Table 10-25. DST\_WRAP\_SIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	WRAPSIZE	R/W	0h	These bits specify the number of bursts to transfer before the destination address wraps around to the beginning address. The actual number is equal to WRAPSIZE + 1. To disable the wrapping function, set WRAPSIZE to a value larger than TRANSFERSIZE. Reset type: SYSRSn

**10.9.3.15 DST\_WRAP\_COUNT Register (Offset = Eh) [reset = 0h]**

DST\_WRAP\_COUNT is shown in [Figure 10-26](#) and described in [Table 10-26](#).

Return to the [Summary Table](#).

Destination Wrap Count Register

**Figure 10-26. DST\_WRAP\_COUNT Register**

15	14	13	12	11	10	9	8
WRAPSIZE							
R/W-0h							
7	6	5	4	3	2	1	0
WRAPSIZE							
R/W-0h							

**Table 10-26. DST\_WRAP\_COUNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	WRAPSIZE	R/W	0h	These bits indicate the number of bursts left before wrapping the destination address. Reset type: SYSRSn

**10.9.3.16 DST\_WRAP\_STEP Register (Offset = Fh) [reset = 0h]**

DST\_WRAP\_STEP is shown in [Figure 10-27](#) and described in [Table 10-27](#).

Return to the [Summary Table](#).

Destination Wrap Step Register

**Figure 10-27. DST\_WRAP\_STEP Register**

15	14	13	12	11	10	9	8
WRAPSTEP							
R/W-0h							
7	6	5	4	3	2	1	0
WRAPSTEP							
R/W-0h							

**Table 10-27. DST\_WRAP\_STEP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	WRAPSTEP	R/W	0h	<p>These bits specify the change in the destination beginning address when the wrap counter reaches zero. The size must be a 16-bit two's complement value between -4096 and 4095 (inclusive). This value is added to the destination address when wrapping occurs.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No address change            1h (R/W) = Add 1 to the address            2h (R/W) = Add 2 to the address            FFEh (R/W) = Add 4094 to the address            FFFh (R/W) = Add 4095 to the address            F00h (R/W) = Subtract 4096 from the address            F001h (R/W) = Subtract 4095 from the address            FFFEh (R/W) = Subtract 2 from the address            FFFFh (R/W) = Subtract 1 from the address</p>

**10.9.3.17 SRC\_BEG\_ADDR\_SHADOW Register (Offset = 10h) [reset = 0h]**

SRC\_BEG\_ADDR\_SHADOW is shown in [Figure 10-28](#) and described in [Table 10-28](#).

Return to the [Summary Table](#).

Source Begin Address Shadow Register

**Figure 10-28. SRC\_BEG\_ADDR\_SHADOW 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
BEGADDR																															
R/W-0h																															

**Table 10-28. SRC\_BEG\_ADDR\_SHADOW Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BEGADDR	R/W	0h	Shadow Source Beginning Address  At the start of a transfer, the value in this register is loaded into the SRC_BEG_ADDR_ACTIVE register and used as the beginning value for the source address. This register can be safely updated during a transfer.  Reset type: SYSRSn

**10.9.3.18 SRC\_ADDR\_SHADOW Register (Offset = 12h) [reset = 0h]**

SRC\_ADDR\_SHADOW is shown in [Figure 10-29](#) and described in [Table 10-29](#).

Return to the [Summary Table](#).

Source Address Shadow Register

**Figure 10-29. SRC\_ADDR\_SHADOW 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/W-0h																															

**Table 10-29. SRC\_ADDR\_SHADOW Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	0h	Shadow Source Address  At the start of a transfer, the value in this register is loaded into the SRC_ADDR_ACTIVE register and used as the value of the source address. This register can be safely updated during a transfer. Reset type: SYSRSn

**10.9.3.19 SRC\_BEG\_ADDR\_ACTIVE Register (Offset = 14h) [reset = 0h]**

SRC\_BEG\_ADDR\_ACTIVE is shown in [Figure 10-30](#) and described in [Table 10-30](#).

Return to the [Summary Table](#).

Source Begin Address Active Register

**Figure 10-30. SRC\_BEG\_ADDR\_ACTIVE 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
BEGADDR																															
R/W-0h																															

**Table 10-30. SRC\_BEG\_ADDR\_ACTIVE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BEGADDR	R/W	0h	Active Source Beginning Address  If a transfer is ongoing, this register holds the current beginning value for the source address. This address may be updated after wrapping.  When a transfer starts, this register is loaded with the shadow address from the SRC_BEG_ADDR_SHADOW register. Reset type: SYSRSn

**10.9.3.20 SRC\_ADDR\_ACTIVE Register (Offset = 16h) [reset = 0h]**

SRC\_ADDR\_ACTIVE is shown in [Figure 10-31](#) and described in [Table 10-31](#).

Return to the [Summary Table](#).

Source Address Active Register

**Figure 10-31. SRC\_ADDR\_ACTIVE 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/W-0h																															

**Table 10-31. SRC\_ADDR\_ACTIVE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	0h	<p>Active Source Address</p> <p>If a transfer is ongoing, this register holds the current value of the source address. This address may change after a write, a burst, or wrapping.</p> <p>Reset type: SYSRSn</p>

**10.9.3.21 DST\_BEG\_ADDR\_SHADOW Register (Offset = 18h) [reset = 0h]**

DST\_BEG\_ADDR\_SHADOW is shown in [Figure 10-32](#) and described in [Table 10-32](#).

Return to the [Summary Table](#).

Destination Begin Address Shadow Register

**Figure 10-32. DST\_BEG\_ADDR\_SHADOW 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
BEGADDR																															
R/W-0h																															

**Table 10-32. DST\_BEG\_ADDR\_SHADOW Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BEGADDR	R/W	0h	Shadow Destination Beginning Address  At the start of a transfer, the value in this register is loaded into the DST_BEG_ADDR_ACTIVE register and used as the beginning value for the destination address. This register can be safely updated during a transfer.  Reset type: SYSRSn



**10.9.3.22 DST\_ADDR\_SHADOW Register (Offset = 1Ah) [reset = 0h]**

DST\_ADDR\_SHADOW is shown in [Figure 10-33](#) and described in [Table 10-33](#).

Return to the [Summary Table](#).

Destination Address Shadow Register

**Figure 10-33. DST\_ADDR\_SHADOW 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/W-0h																															

**Table 10-33. DST\_ADDR\_SHADOW Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	0h	Shadow Destination Address  At the start of a transfer, the value in this register is loaded into the DST_ADDR_ACTIVE register and used as the value of the destination address. This register can be safely updated during a transfer.  Reset type: SYSRSn

**10.9.3.23 DST\_BEG\_ADDR\_ACTIVE Register (Offset = 1Ch) [reset = 0h]**

DST\_BEG\_ADDR\_ACTIVE is shown in [Figure 10-34](#) and described in [Table 10-34](#).

Return to the [Summary Table](#).

Destination Begin Address Active Register

**Figure 10-34. DST\_BEG\_ADDR\_ACTIVE 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
BEGADDR																															
R/W-0h																															

**Table 10-34. DST\_BEG\_ADDR\_ACTIVE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BEGADDR	R/W	0h	Active Destination Beginning Address  If a transfer is ongoing, this register holds the current destination value for the source address. This address may be updated after wrapping.  When a transfer starts, this register is loaded with the shadow address from the DST_BEG_ADDR_SHADOW register.  Reset type: SYSRSn

**10.9.3.24 DST\_ADDR\_ACTIVE Register (Offset = 1Eh) [reset = 0h]**

DST\_ADDR\_ACTIVE is shown in [Figure 10-35](#) and described in [Table 10-35](#).

Return to the [Summary Table](#).

Destination Address Active Register

**Figure 10-35. DST\_ADDR\_ACTIVE 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/W-0h																															

**Table 10-35. DST\_ADDR\_ACTIVE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	0h	<p>Active Destination Address</p> <p>If a transfer is ongoing, this register holds the current value of the destination address. This address may change after a write, a burst, or wrapping.</p> <p>Reset type: SYSRSn</p>

## 10.9.4 Register to Driverlib Function Mapping

**Table 10-36. DMA Registers to Driverlib Functions**

File	Driverlib Function
<b>CTRL</b>	
dma.h	DMA_initController
<b>DEBUGCTRL</b>	
dma.h	DMA_setEmulationMode
<b>PRIORITYCTRL1</b>	
dma.h	DMA_setPriorityMode
<b>MODE</b>	
dma.c	DMA_configMode
dma.h	DMA_enableTrigger
dma.h	DMA_disableTrigger
dma.h	DMA_enableInterrupt
dma.h	DMA_disableInterrupt
dma.h	DMA_enableOverrunInterrupt
dma.h	DMA_disableOverrunInterrupt
dma.h	DMA_setInterruptMode
<b>CONTROL</b>	
dma.h	DMA_forceTrigger
dma.h	DMA_clearTriggerFlag
dma.h	DMA_getTriggerFlagStatus
dma.h	DMA_startChannel
dma.h	DMA_stopChannel
dma.h	DMA_clearErrorFlag
<b>BURST_SIZE</b>	
dma.c	DMA_configBurst
<b>SRC_BURST_STEP</b>	
dma.c	DMA_configBurst
<b>DST_BURST_STEP</b>	
dma.c	DMA_configBurst
<b>TRANSFER_SIZE</b>	
dma.c	DMA_configTransfer
<b>SRC_TRANSFER_STEP</b>	
dma.c	DMA_configTransfer
<b>DST_TRANSFER_STEP</b>	
dma.c	DMA_configTransfer
<b>SRC_WRAP_SIZE</b>	
dma.c	DMA_configWrap
<b>SRC_WRAP_STEP</b>	
dma.c	DMA_configWrap
<b>DST_WRAP_SIZE</b>	
dma.c	DMA_configWrap
<b>DST_WRAP_STEP</b>	
dma.c	DMA_configWrap
<b>SRC_BEG_ADDR_SHADOW</b>	
dma.c	DMA_configAddresses
dma.h	DMA_configSourceAddress

**Table 10-36. DMA Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>SRC_ADDR_SHADOW</b>	
dma.c	DMA_configAddresses
dma.h	DMA_configSourceAddress
<b>DST_BEG_ADDR_SHADOW</b>	
dma.c	DMA_configAddresses
dma.h	DMA_configDestAddress
<b>DST_ADDR_SHADOW</b>	
dma.c	DMA_configAddresses
dma.h	DMA_configDestAddress

## ***Embedded Real-time Analysis and Diagnostic (ERAD)***

---



---

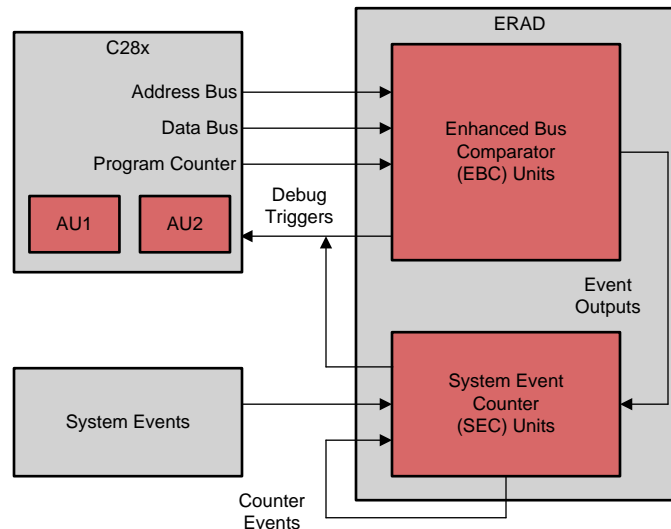
This chapter describes the features and operation of the embedded real-time analysis and diagnostic (ERAD) module. The ERAD module enhances the debug and system analysis capabilities of the device. The debug and system analysis enhancements provided by the ERAD module are implemented external to the CPU. The ERAD module consists of the enhanced bus comparator (EBC) units and the system event counter (SEC) units. The EBC units are used to generate hardware breakpoints, hardware watch points, and other output events. The SEC units are used to analyze and profile the system. The ERAD module is accessible both by the debugger and the application software, which significantly increases the debug capabilities of many real-time systems, especially in situations where the debugger is not connected.

Topic	Page
<b>11.1 Introduction .....</b>	<b>1415</b>
<b>11.2 Enhanced Bus Comparator Unit .....</b>	<b>1415</b>
<b>11.3 System Event Counter Unit .....</b>	<b>1416</b>
<b>11.4 ERAD Ownership, Initialization and Reset.....</b>	<b>1419</b>
<b>11.5 ERAD Programming Sequence .....</b>	<b>1419</b>
<b>11.6 ERAD Registers .....</b>	<b>1422</b>

## 11.1 Introduction

The ERAD module is shown in [Figure 11-1](#).

**Figure 11-1. ERAD Overview**



ERAD enhances the debug and system analysis capabilities of the device external to the CPU. The C28x CPU alone has two analysis resources; Analysis Unit 1 (AU1) and Analysis Unit 2 (AU2). The first analysis unit counts events or monitors address buses. The second analysis unit monitors address and data buses. The two analysis units can be configured for hardware breakpoints or hardware watch points, and additionally the first analysis unit can be configured as a benchmark counter or event counter. The ERAD module further expands this capability to provide additional hardware breakpoints, hardware watch points, and counters for profiling, as well as other advanced features. The ERAD module can be utilized by the debugger, and also by the application software. For many real-time systems, it is not always possible to connect a debugger and perform an intrusive debug. Under these situations, the user's code has the ability to set up and control the ERAD module in order to debug and profile the system without disturbing the end application.

The ERAD module consists of eight enhanced bus comparator (EBC) units and four system event counter (SEC) units. The EBC units monitor buses and generate output events. The SEC units can be used with EBC units to profile and analyze the system. These units are described in detail in the following sections.

## 11.2 Enhanced Bus Comparator Unit

The EBC units connect to the CPU similar to any standard direct memory interface connection. The program space buses, data space buses, debug qualifiers for memory access, and debug-related strobes necessary for breakpoints, watch points and trace points, are connected between the CPU and these units. The EBC units are usually controlled by the debug software. However, a CPU application code can also configure and use these units for interrupt and event generation. The ERAD module can be configured and used by either the application software or debug software. It is not possible for an EBC unit to be used by both the application software and the debug software simultaneously. The ERAD module will be owned by either the debugger or the application software through the memory-mapped registers.

The EBC unit has the following capabilities:

- Generate hardware breakpoints
- Generate hardware watch points
- Generate trace tags for instruction fetch matches and generate RTOSINT
- Monitor data read address buses, data write address buses, data write data bus, and generate RTOSINT
- Generate an event output which can be used by other modules. This is done through monitoring any of the program address buses, Virtual Program Counter (VPC), or the Program Counter of the CPU

The following features are not supported by the EBC units:

- Chain breakpoints
- Ability to monitor DMA transfers
- Ability to monitor CLA buses

The EBC units have the capability to monitor a range of addresses by defining masks and generating outputs based on greater than, less than, or equal events.

The EBC units can be used with the SEC units for profiling and analysis purposes.

### 11.2.1 *Enhanced Bus Comparator Unit Operations*

The following operations are supported by the EBC unit:

- Operation of a hardware breakpoint: The EBC unit will generate the appropriate tags for an access made to a particular address. When the instruction is going to enter the DECODE 2 (D2) phase of the pipeline, the CPU is halted.
- Operation of a watch point: Unlike breakpoints which halt the CPU before a designated address is executed, watch points will detect the occurrence of a data memory read or write, and will halt the CPU. There is no precise definition as to when the CPU will halt, since it is entirely dependant on the current state of the CPU pipeline. The CPU will halt at the next interruptible boundary.
- Operation of program trace: This operation is very similar to the hardware breakpoint operation. The difference between the CPU behavior is that it will internally generate an RTOSINT when the tagged instruction reaches the DECODE 2 (D2) phase of the pipeline. If this instruction is discarded in the fetch buffer due to discontinuity, then no RTOSINT will be generated.
- Operation of RTOSINT<sub>n</sub>: This is an alternative to a watch point where an interrupt can be generated to the CPU on detecting the designated read and write access.

It is important to note that a hardware breakpoint will only halt the CPU if a debugger is connected.

### 11.3 **System Event Counter Unit**

The SEC units provide better system profiling, analysis, and debug capability. The SEC units contain counters which can enhance the debug and profiling process in various types of system scenarios such as:

- Profiling code segments
- Counting duration between specified memory reads and writes
- Counting system events (such as interrupts)
- Counting duration between system events
- System timer
- Measuring the number of wait states in code segments
- Measuring the maximum amount of time spent in between a pair of events, measured over multiple iterations
- Chaining counters in order to link events or create larger counters

Furthermore, the SEC unit has the capability to:

- Function as a counter capable of counting:
  - Any of the match events generated by the EBC units
  - Events generated by the EBC units. These events can be used to start and stop the counting.
  - System events including the PIE interrupts, timer interrupts, and CLA task interrupts. These system events can be used to start and stop the counting.
  - More information on the sources of the SEC units can be found at the Event Selector Mux Signals Table.
- Generate an interrupt or a watch point if the count reaches a reference value.
- Operate in counting mode in one of the following two modes:
  - Duration mode: The counter will count the CPU cycles as long as the event is active.



- Event mode: The counter will count only the positive edge of the event signal. This is effectively counting the number of times the event transitions from inactive to active.

### 11.3.1 System Event Counter Modes

The following are the key features of the SEC unit. The counters are initialized to zero and will always count up.

- Continuous Count: In this mode the counter continues to count as specified by the input selector. If the counter reaches the maximum value, it resets to zero and will continue to count up. A sticky overflow bit will be set to indicate that this counter had overflowed. The counter can count the CPU cycles without any events selected. In this mode the module can be used as a software controlled SYSCLK counter.
- Timer Mode Count: In this mode, the counter counts up to a set reference value and after reaching the reference value, it resets to zero. The counter will generate an event which can send an interrupt to the CPU or generate a watch point. The counter can be set up to either continue incrementing or reset when a match event occurs.
- Start-Stop Count: In this mode, two events are allowed to act as start and stop indicators to the counter. The counter will commence counting only when the defined start event occurs. The counter will then continue to count up until the stop event occurs. Once the first start event has occurred, further start events will be ignored till the stop event occurs.

#### 11.3.1.1 Counting Active Levels vs. Edges

The SEC units can be configured to either count active levels or edges of the selected inputs.

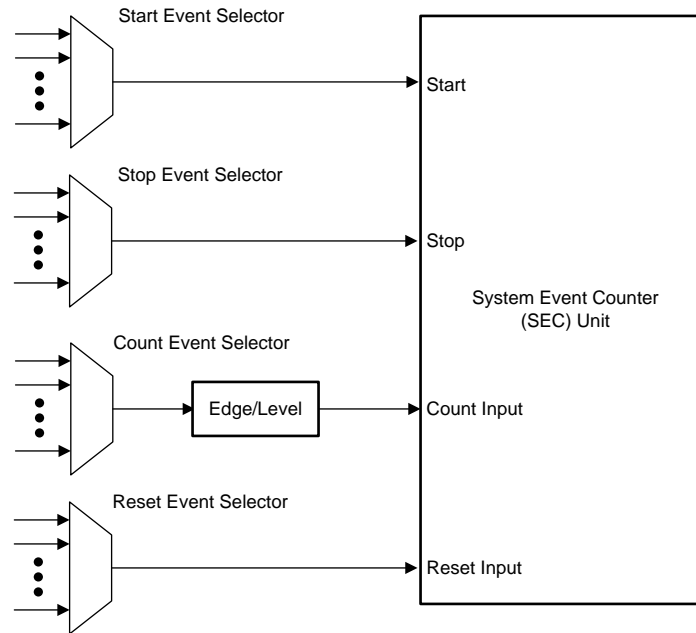
Each SEC unit has eight inputs from the EBC units and many inputs from other events in the device. Each SEC unit can be configured to count any of the input events or just count up on every cycle. For example if an input event occurs and is active for 25 cycles, the SEC unit's counter will increment only by 1 in event mode, whereas in the duration mode, it will increment by 25.

#### 11.3.1.2 Max mode

Max mode is also supported by the SEC units. This mode allows the user to detect the maximum count that has occurred during various count iterations in start-stop mode. For example, a user can set up the counter in the start-stop count mode to count the duration of a critical code loop. Every time the stop event occurs and the counter stops, the counter value is checked against the current MAX\_COUNT present in the register. If the new value is greater, then the MAX\_COUNT register is updated. The counter will always reset to zero at the stop event and will be ready to start counting on the next start event. Therefore the MAX\_COUNT will contain the maximum number of cycles that occurred between the start and stop condition over many iterations.

#### 11.3.1.3 Input Signal Selection

The SEC inputs can be selected from various signals from in the system to enable debug and system analysis. [Figure 11-2](#) shows the SEC inputs.

**Figure 11-2. System Event Counter Inputs**


Each event selector MUX can select from various signals on in the system. These signals are show in [Table 11-1](#) .

**Table 11-1. Event Selector Mux Signals**

CTM_INP_SEL, STA_INP_SEL, STO_INP_SEL, RST_INP_SEL	EVENT_INPUT_SELECTED
INP_SEL[0]	HWBP1
INP_SEL[1]	HWBP2
INP_SEL[2]	HWBP3
INP_SEL[3]	HWBP4
INP_SEL[4]	HWBP5
INP_SEL[5]	HWBP6
INP_SEL[6]	HWBP7
INP_SEL[7]	HWBP8
INP_SEL[8]	COUNTER1_EVENT
INP_SEL[9]	COUNTER2_EVENT
INP_SEL[10]	COUNTER3_EVENT
INP_SEL[11]	COUNTER4_EVENT
INP_SEL[12]	PIE_INT1
INP_SEL[13]	PIE_INT2
INP_SEL[14]	PIE_INT3
INP_SEL[15]	PIE_INT4
INP_SEL[16]	PIE_INT5
INP_SEL[17]	PIE_INT6
INP_SEL[18]	PIE_INT7
INP_SEL[19]	PIE_INT8
INP_SEL[20]	PIE_INT9
INP_SEL[21]	PIE_INT10
INP_SEL[22]	PIE_INT11
INP_SEL[23]	PIE_INT12

**Table 11-1. Event Selector Mux Signals (continued)**

CTM_INP_SEL, STA_INP_SEL, STO_INP_SEL, RST_INP_SEL	EVENT_INPUT_SELECTED
INP_SEL[24]	TIMER1_TINT1
INP_SEL[25]	TIMER2_TINT2
INP_SEL[26]	CLA_INTERRUPT1
INP_SEL[27]	CLA_INTERRUPT2
INP_SEL[28]	CLA_INTERRUPT3
INP_SEL[29]	CLA_INTERRUPT4
INP_SEL[30]	CLA_INTERRUPT5
INP_SEL[31]	CLA_INTERRUPT8

### 11.3.2 Reset on Event

It is also possible to reset the counters on external events. Additionally all the counter event outputs are fed back to each of the counter's input MUX which will select the event that can be used as a reset input. When enabled, an active high on the reset input will cause the counter to reset. This gives a powerful feature which allows the user to set up threshold monitors. This can be used to flag an interrupt or a watch point if the distance between two events crosses a certain threshold.

### 11.3.3 Operation Conditions

The SEC units will count accurately only when the CPU is operating in normal conditions. If the counters are running and capturing the CPU cycles while the CPU is controlled through the debugger in order to single-step through the code, then the result may differ from when the CPU was executing the code in normal conditions.

It is also important to note that if the counters are set up to use the value of the raw program counter register (VPC) as the source for the start and stop events, the value of the CPU cycles measured may be off by a few cycles when the CALL instruction is executed.

## 11.4 ERAD Ownership, Initialization and Reset

Even though the features of this module are meant for the debugger to use, the user application may also need to take advantage of the capabilities to monitor the buses and generate interrupts and events. It is an option to completely hand over the ownership of the ERAD module to the application software or the debugger. Another option is for both the application code and the debugger to use the ERAD module. Only the owner of the module (application code or debugger) is allowed to use the module at that time. The responsibility of resolving any ownership conflict, is on the software. The last option is to have no owner. In this mode, both the application code and the debugger have the capability to access the module at the same time. It is critical for the software, both on the application side and the debugger side, to resolve any conflicts. In this mode, it is possible for the debugger to use some of the EBC units and the SEC units, while the application software uses the remaining units.

The ERAD module will initialize its internal states and all the registers to their initialized states under the following conditions:

- At power-on-reset (POR)
- With DCON and SYSRSN
  - Debug logic disconnected when the debugger owns the module
  - Functional reset when application owns the module

## 11.5 ERAD Programming Sequence

The ERAD module can be used to set hardware breakpoints and hardware watch points. The programming sequence to set hardware breakpoints, hardware watch points, or to use the timers to profile and analyze the system are described in this section. The same sequence can be used through the debugger software or the application code.

Please refer to the Driverlib example projects in C2000Ware for JavaScript files to configure the ERAD module. Example projects are also available to showcase the usage of these script files. These examples are included in the latest Driverlib install.

<C2000Ware installation>\driverlib\28004x\examples\erad

### 11.5.1 **Hardware Breakpoint and Hardware Watch Point Programming Sequence**

The programming sequence is identical when using the EBC units, regardless of whether it is a debug software or the application programming the units. A typical programming sequence for a unit is described below.

- Read and make sure the ownership is set as expected; if not, acquire the ownership before proceeding further if required.
- Ensure the unit is in IDLE mode.
- Set up the address reference, mask, bus select and stop bits.
- Enable the corresponding module bit in the global enable register.
- Once the usage is completed, write '1' to Clear EVENT\_FIRED sticky bit; This will take the module back to the enabled state.

The example programming sequences for hardware breakpoints and hardware watch points are shown below.

Set a hardware breakpoint on address 0x201000:

- Read GLBL\_OWNER to confirm ownership.
- Read HWBP\_STATUS to confirm the module is in IDLE state.
- Write 0x0 to HWBP\_MASK.
- Write 0x201000 to HWBP\_REF.
- Write 0x20 to HWBP\_CNTL (STOP = 1, BUS\_SEL = 000/PAB). If a trace is to be generated instead of a breakpoint, then set HWBP\_CNTL to 0x40 instead of (STOP = 0).
- Enable the corresponding module bit in the global enable register.

Set a hardware watch point on read of addresses from 0x121010 to 0x12101f:

- Read GLBL\_OWNER to confirm ownership.
- Read HWBP\_STATUS to confirm the module is in IDLE state.
- Write 0xf to HWBP\_MASK.
- Write 0x121010 to HWBP\_REF.
- Write 0x2C to HWBP\_CNTL (STOP = 1, BUS\_SEL = 011/DRAB). If an RTOSINTn is to be generated instead of a watch point, then set HWBP\_CNTL to 0x4C instead (STOP = 0).
- Enable the corresponding module bit in the global enable register.

Set a hardware watch point on write to address 0xff10101a:

- Read GLBL\_OWNER to confirm ownership
- Read HWBP\_STATUS to confirm the module is in IDLE state
- Write 0x0 to HWBP\_MASK
- Write 0xff10101a to HWBP\_REF
- Write 0x28 to HWBP\_CNTL (STOP = 1, BUS\_SEL = 010/DWAB). If an RTOSINTn is to be generated instead of a watch point, then set HWBP\_CNTL to 0x48 instead (STOP = 0)
- Enable the corresponding module bit in the global enable register.

### 11.5.2 **Timer and Counter Programming Sequence**

The programming sequence is identical when using the SEC units, regardless of whether it is a debug software, or the application, programming the units. Typical programming sequence for a unit is described below.

- Read and make sure the ownership is set as expected. If not, acquire the ownership before proceeding

further if required.

- Ensure the unit is in IDLE mode.
- Set up the counter reference, counter registers (clear/reset if a clean start is required) and CTM\_CNTL.
- Enable the corresponding module bit in the global enable register.
- Once the usage is completed, write '1' to Clear EVENT\_FIRED sticky bit. This will take the module back to the enabled state.

Set up a free running counter:

- Read and make sure the ownership is set as expected. If not, acquire the ownership before proceeding further, if required.
- Read CTM\_STATUS to confirm that the module is in IDLE state.
- Write 0x0 to CTM\_INPUT\_SEL.
- Write 0x0 to CTM\_CNTL.
- Enable the module in the GLBL\_ENABLE register.

Set up the counter to count the duration spent between addresses 0x1000 and 0x1210:

- Read and make sure the ownership is set as expected. If not, acquire the ownership before proceeding further, if required.
- Read CTM\_STATUS to confirm that the module is in IDLE state.
- Set up the EBC unit 1 to generate an event for VPC = 0x1000.
- Set up the EBC unit 2 to generate an event for VPC = 0x1210.
- Set up the start and stop input selects to use comparator event 1 and 2, respectively. This is achieved by writing the value 0x0800 to CTM\_INPUT\_SEL register (bits 15:11 = 0b00001 and bits 10:6 = 0b00000 and all other bits are set to 0).
- Enable the counter in the START\_STOP\_MODE of operation. This is achieved by writing 0x4 to CTM\_CNTL.
- Enable the module in the GLBL\_ENABLE register.

Set up the counter to count the number of times a function at address 0x2010 is called and fire an RTOSINT if this count reaches 0x300:

- Read and make sure the ownership is set as expected. If not, acquire the ownership before proceeding further, if required.
- Read CTM\_STATUS to confirm that the module is in IDLE state.
- Set up the EBC unit 1 to generate an event for VPC = 0x2010.
- Setup the counter to select comparator event 1 as the event to count. This is achieved by writing the value 0x0001 to CTM\_INPUT\_SEL register (Register bits 5:1 = 0b00000 and bit 0 = 1).
- Write 0x300 CTM\_REF.
- Enable the counter in the EVENT\_MODE, and also allow it to generate an RTOSINT when the count matches the reference. This is achieved by writing 0x88 to CTM\_CNTL (bit 3 = 1 and bit 7 = 1).
- Enable the module in the GLBL\_ENABLE register.

## 11.6 ERAD Registers

This section describes the Embedded Real-Time Analysis and Diagnostic Registers.

### 11.6.1 ERAD Base Addresses

**Table 11-2. ERAD Base Address Table**

Device Registers	Register Name	Start Address	End Address
EradGlobalRegs	ERAD_GLOBAL_REGS	0x0005_E800	0x0005_E812
EradHWBP1Regs	ERAD_HWBP_REGS	0x0005_E900	0x0005_E907
EradHWBP2Regs	ERAD_HWBP_REGS	0x0005_E908	0x0005_E90F
EradHWBP3Regs	ERAD_HWBP_REGS	0x0005_E910	0x0005_E917
EradHWBP4Regs	ERAD_HWBP_REGS	0x0005_E918	0x0005_E91F
EradHWBP5Regs	ERAD_HWBP_REGS	0x0005_E920	0x0005_E927
EradHWBP6Regs	ERAD_HWBP_REGS	0x0005_E928	0x0005_E92F
EradHWBP7Regs	ERAD_HWBP_REGS	0x0005_E930	0x0005_E937
EradHWBP8Regs	ERAD_HWBP_REGS	0x0005_E938	0x0005_E93F
EradCounter1Regs	ERAD_COUNTER_REGS	0x0005_E980	0x0005_E98F
EradCounter2Regs	ERAD_COUNTER_REGS	0x0005_E990	0x0005_E99F
EradCounter3Regs	ERAD_COUNTER_REGS	0x0005_E9A0	0x0005_E9AF
EradCounter4Regs	ERAD_COUNTER_REGS	0x0005_E9B0	0x0005_E9BF

## 11.6.2 ERAD\_GLOBAL\_REGS Registers

Table 11-3 lists the ERAD\_GLOBAL\_REGS registers. All register offset addresses not listed in Table 11-3 should be considered as reserved locations and the register contents should not be modified.

**Table 11-3. ERAD\_GLOBAL\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	GLBL_EVENT_STAT	Global Event Status Register		<a href="#">Go</a>
2h	GLBL_HALT_STAT	Global Halt Status Register		<a href="#">Go</a>
4h	GLBL_ENABLE	Global Enable Register	EALLOW	<a href="#">Go</a>
6h	GLBL_CTM_RESET	Global Counter Reset	EALLOW	<a href="#">Go</a>
Ah	GLBL_OWNER	Global Ownership	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 11-4 shows the codes that are used for access types in this section.

**Table 11-4. ERAD\_GLOBAL\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 11.6.2.1 GLBL\_EVENT\_STAT Register (Offset = 0h) [reset = 0h]

GLBL\_EVENT\_STAT is shown in [Figure 11-3](#) and described in [Table 11-5](#).

Return to the [Summary Table](#).

This register contains one bit for each of the bus comparator modules and the counter modules that are present in a device. Each bit directly reflects the state of the EVENT\_FIRED bit of the corresponding module. This facilitates software to just read one register and find out if any of the debug modules had fired.

**Figure 11-3. GLBL\_EVENT\_STAT Register**

15	14	13	12	11	10	9	8
RESERVED				CTM4	CTM3	CTM2	CTM1
R-0h				R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
HWBP8	HWBP7	HWBP6	HWBP5	HWBP4	HWBP3	HWBP2	HWBP1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 11-5. GLBL\_EVENT\_STAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	CTM4	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Counter unit 4.  0 No Event 1 Event Fired Reset type: ERAD_RESET
10	CTM3	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Counter unit 3.  0 No Event 1 Event Fired Reset type: ERAD_RESET
9	CTM2	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Counter unit 2.  0 No Event 1 Event Fired Reset type: ERAD_RESET
8	CTM1	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Counter unit 1.  0 No Event 1 Event Fired Reset type: ERAD_RESET
7	HWBP8	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 8.  0 No Event 1 Event Fired Reset type: ERAD_RESET
6	HWBP7	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 7.  0 No Event 1 Event Fired Reset type: ERAD_RESET



**Table 11-5. GLBL\_EVENT\_STAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	HWBP6	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 6.  0 No Event 1 Event Fired Reset type: ERAD_RESET
4	HWBP5	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 5.  0 No Event 1 Event Fired Reset type: ERAD_RESET
3	HWBP4	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 4.  0 No Event 1 Event Fired Reset type: ERAD_RESET
2	HWBP3	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 3.  0 No Event 1 Event Fired Reset type: ERAD_RESET
1	HWBP2	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 2.  0 No Event 1 Event Fired Reset type: ERAD_RESET
0	HWBP1	R	0h	This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 1.  0 No Event 1 Event Fired Reset type: ERAD_RESET

### 11.6.2.2 GLBL\_HALT\_STAT Register (Offset = 2h) [reset = 0h]

GLBL\_HALT\_STAT is shown in [Figure 11-4](#) and described in [Table 11-6](#).

Return to the [Summary Table](#).

This register contains one bit for each of the bus comparator modules and the counter modules that are present in a device. Each bit directly reflects the state of the Completed status bit. This facilitates software to just read one register and find out if any of the debug modules have fired.

**Figure 11-4. GLBL\_HALT\_STAT Register**

15	14	13	12	11	10	9	8
RESERVED				CTM4	CTM3	CTM2	CTM1
R-0h				R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
HWBP8	HWBP7	HWBP6	HWBP5	HWBP4	HWBP3	HWBP2	HWBP1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 11-6. GLBL\_HALT\_STAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	CTM4	R	0h	This bit directly reflects the state of the completed bit of the Counter unit 4.  0 No Event 1 Event Fired Reset type: ERAD_RESET
10	CTM3	R	0h	This bit directly reflects the state of the completed bit of the Counter unit 3.  0 No Event 1 Event Fired Reset type: ERAD_RESET
9	CTM2	R	0h	This bit directly reflects the state of the completed bit of the Counter unit 2.  0 No Event 1 Event Fired Reset type: ERAD_RESET
8	CTM1	R	0h	This bit directly reflects the state of the completed bit of the Counter unit 1.  0 No Event 1 Event Fired Reset type: ERAD_RESET
7	HWBP8	R	0h	This bit directly reflects the state of the completed bit of the Bus Comparator unit 8.  0 Not Completed 1 Completed Reset type: ERAD_RESET
6	HWBP7	R	0h	This bit directly reflects the state of the completed bit of the Bus Comparator unit 7.  0 Not Completed 1 Completed Reset type: ERAD_RESET

**Table 11-6. GLBL\_HALT\_STAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	HWBP6	R	0h	This bit directly reflects the state of the completed bit of the Bus Comparator unit 6.  0 Not Completed 1 Completed Reset type: ERAD_RESET
4	HWBP5	R	0h	This bit directly reflects the state of the completed bit of the Bus Comparator unit 5.  0 Not Completed 1 Completed Reset type: ERAD_RESET
3	HWBP4	R	0h	This bit directly reflects the state of the completed bit of the Bus Comparator unit 4.  0 Not Completed 1 Completed Reset type: ERAD_RESET
2	HWBP3	R	0h	This bit directly reflects the state of the completed bit of the Bus Comparator unit 3.  0 Not Completed 1 Completed Reset type: ERAD_RESET
1	HWBP2	R	0h	This bit directly reflects the state of the completed bit of the Bus Comparator unit 2.  0 Not Completed 1 Completed Reset type: ERAD_RESET
0	HWBP1	R	0h	This bit directly reflects the state of the completed bit of the Bus Comparator unit 1.  0 Not Completed 1 Completed Reset type: ERAD_RESET

### 11.6.2.3 GLBL\_ENABLE Register (Offset = 4h) [reset = 0h]

GLBL\_ENABLE is shown in [Figure 11-5](#) and described in [Table 11-7](#).

Return to the [Summary Table](#).

This register contains one bit for each of the bus comparator modules and the counter modules that are present in a device. Each bit directly acts as a global enable for the corresponding module. This bit has to be set to 1 for the module to be functional.

**Figure 11-5. GLBL\_ENABLE Register**

15	14	13	12	11	10	9	8
RESERVED				CTM4	CTM3	CTM2	CTM1
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
HWBP8	HWBP7	HWBP6	HWBP5	HWBP4	HWBP3	HWBP2	HWBP1
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 11-7. GLBL\_ENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	CTM4	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Counter unit 4.  0 Disabled 1 Enabled Reset type: ERAD_RESET
10	CTM3	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Counter unit 3.  0 Disabled 1 Enabled Reset type: ERAD_RESET
9	CTM2	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Counter unit 2.  0 Disabled 1 Enabled Reset type: ERAD_RESET
8	CTM1	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Counter unit 1.  0 Disabled 1 Enabled Reset type: ERAD_RESET
7	HWBP8	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 8.  0 Disabled 1 Enabled Reset type: ERAD_RESET
6	HWBP7	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 7.  0 Disabled 1 Enabled Reset type: ERAD_RESET

**Table 11-7. GLBL\_ENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	HWBP6	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 6.  0 Disabled 1 Enabled Reset type: ERAD_RESET
4	HWBP5	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 5.  0 Disabled 1 Enabled Reset type: ERAD_RESET
3	HWBP4	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 4.  0 Disabled 1 Enabled Reset type: ERAD_RESET
2	HWBP3	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 3.  0 Disabled 1 Enabled Reset type: ERAD_RESET
1	HWBP2	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 2.  0 Disabled 1 Enabled Reset type: ERAD_RESET
0	HWBP1	R/W	0h	This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 1.  0 Disabled 1 Enabled Reset type: ERAD_RESET

### 11.6.2.4 GLBL\_CTM\_RESET Register (Offset = 6h) [reset = 0h]

GLBL\_CTM\_RESET is shown in [Figure 11-6](#) and described in [Table 11-8](#).

Return to the [Summary Table](#).

This register contains one bit for each of the counter modules that are present in a device. Each bit directly acts as a reset for the counters for the corresponding module. (It does not affect anything else except resetting the counter.

Example: If the counter was previously incrementing before reset, then on a reset event the counter gets reset and continues to increment again).

**Figure 11-6. GLBL\_CTM\_RESET Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				CTM4	CTM3	CTM2	CTM1
R-0h				R-0/W-0h	R-0/W-0h	R-0/W-0h	R-0/W-0h

**Table 11-8. GLBL\_CTM\_RESET Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	CTM4	R-0/W	0h	This bit directly resets the state the Counter unit 4. 0 No Effect 1 Reset Reset type: ERAD_RESET
2	CTM3	R-0/W	0h	This bit directly resets the state the Counter unit 3. 0 No Effect 1 Reset Reset type: ERAD_RESET
1	CTM2	R-0/W	0h	This bit directly resets the state the Counter unit 2. 0 No Effect 1 Reset Reset type: ERAD_RESET
0	CTM1	R-0/W	0h	This bit directly resets the state the Counter unit 1. 0 No Effect 1 Reset Reset type: ERAD_RESET

### 11.6.2.5 GLBL\_OWNER Register (Offset = Ah) [reset = 0h]

GLBL\_OWNER is shown in [Figure 11-7](#) and described in [Table 11-9](#).

Return to the [Summary Table](#).

Global Ownership

**Figure 11-7. GLBL\_OWNER Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						OWNER	
R-0h						R/W-0h	

**Table 11-9. GLBL\_OWNER Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1-0	OWNER	R/W	0h	<p>This register determines whether Application Code or Debugger owns this module or it's kept in No Owner state where debugger or application can access the module.</p> <p>00 No Owner            01 Application owned            10 Debugger owned            11 Reserved</p> <p>Reset type: ERAD_RESET</p>

### 11.6.3 ERAD\_HWBP\_REGS Registers

Table 11-10 lists the ERAD\_HWBP\_REGS registers. All register offset addresses not listed in Table 11-10 should be considered as reserved locations and the register contents should not be modified.

**Table 11-10. ERAD\_HWBP\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	HWBP_MASK	HWBP Mask Register	EALLOW	<a href="#">Go</a>
2h	HWBP_REF	HWBP Reference Register	EALLOW	<a href="#">Go</a>
4h	HWBP_CLEAR	HWBP Clear Register	EALLOW	<a href="#">Go</a>
6h	HWBP_CNTL	HWBP Control Register	EALLOW	<a href="#">Go</a>
7h	HWBP_STATUS	HWBP Status Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 11-11 shows the codes that are used for access types in this section.

**Table 11-11. ERAD\_HWBP\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.



### 11.6.3.1 HWBP\_MASK Register (Offset = 0h) [reset = 0h]

HWBP\_MASK is shown in [Figure 11-8](#) and described in [Table 11-12](#).

Return to the [Summary Table](#).

HWBP Mask Register

**Figure 11-8. HWBP\_MASK 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
MASK																															
R/W-0h																															

**Table 11-12. HWBP\_MASK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	MASK	R/W	0h	<p>This register contains address mask for comparison. The contents of this register are used along with the reference register to determine the address match. The equation used to determine a match is as follows. Match is true if,</p> $(\text{address}   \text{mask}) == (\text{ref}   \text{mask})$ <p>This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored. The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored.</p> <p>Reset type: ERAD_RESET</p>

### 11.6.3.2 HWBP\_REF Register (Offset = 2h) [reset = 0h]

HWBP\_REF is shown in [Figure 11-9](#) and described in [Table 11-13](#).

Return to the [Summary Table](#).

HWBP Reference Register

**Figure 11-9. HWBP\_REF 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
REF																															
R/W-0h																															

**Table 11-13. HWBP\_REF Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	REF	R/W	0h	<p>This register contains the reference address for comparison. The contents of this register are used along with the mask register to determine the address match. The equation used to determine a match is as follows. Match is true if,</p> $(\text{address}   \text{mask}) == (\text{ref}   \text{mask})$ <p>This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored. The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored.</p> <p>Reset type: ERAD_RESET</p>

### 11.6.3.3 HWBP\_CLEAR Register (Offset = 4h) [reset = 0h]

HWBP\_CLEAR is shown in [Figure 11-10](#) and described in [Table 11-14](#).

Return to the [Summary Table](#).

HWBP Clear Register

**Figure 11-10. HWBP\_CLEAR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							EVENT_CLR
R-0h							R-0/W-0h

**Table 11-14. HWBP\_CLEAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	EVENT_CLR	R-0/W	0h	Event Clear register: 0 No action. 1 A write with this bit set to 1 will clear the sticky EVENT_FIRED bit in the HWBP_STATUS register and bring the Breakpoint Module statemachine status back to IDLE. Reads of this bit position will always return a 0. Reset type: ERAD_RESET

### 11.6.3.4 HWBP\_CNTL Register (Offset = 6h) [reset = 0h]

HWBP\_CNTL is shown in [Figure 11-11](#) and described in [Table 11-15](#).

Return to the [Summary Table](#).

HWBP Control Register

**Figure 11-11. HWBP\_CNTL Register**

15	14	13	12	11	10	9	8
RESERVED				RESERVED	RESERVED	COMP_MODE	
R-0h				R/W-0h	R-0h	R/W-0h	
7	6	5	4	3	2	1	0
COMP_MODE	RTOSINT	STOP	BUS_SEL			RESERVED	
R/W-0h	R/W-0h	R/W-0h	R/W-0h			R-0h	

**Table 11-15. HWBP\_CNTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R	0h	Reserved
9-7	COMP_MODE	R/W	0h	Bus comparator compare modes: 000 Regular masked compare HWBP_MSK will be ignored for the following modes: 100 Bus value GT HWBP_REF 101 Bus value GE HWBP_REF 110 Bus value LT HWBP_REF 111 Bus value LE HWBP_REF GT means Greater Than GE means Greater or Equal LT means Less Than LE means Lesser or Equal Reset type: ERAD_RESET
6	RTOSINT	R/W	0h	This bit decides whether the bus comparator unit will generate RTOSINTn interrupt when event matches occur. Note that the event outputs will always be generated regardless of the state of this bit. 0 The bus comparator unit will not cause any action towards the CPU. 1 The bus comparator unit will assert RTOSINTn for matching data accesses and trace tags for matching program fetches. Reset type: ERAD_RESET
5	STOP	R/W	0h	This bit decides whether the bus comparator unit will generate CPU halting signals when event matches occur. Note that the event outputs will always be generated regardless of the state of this bit. 0 The bus comparator unit will not cause any action towards halting the CPU. 1 The bus comparator unit will assert ANASTOP for matching data accesses and break tags for matching program fetches. These can cause the CPU to HALT Reset type: ERAD_RESET

**Table 11-15. HWBP\_CNTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-2	BUS_SEL	R/W	0h	<p>These bits are used to select which CPU buses will be used for comparison to generate the match events. For each bus selected, the corresponding strobes will automatically be selected to determine valid accesses.</p> <p>000 PAB for instruction fetches            011 DRAB for data read accesses            010 DWAB for data write accesses            001 VPC for Program counter match            100 DWDB for write data match            All other combinations are RESERVED.            Reset type: ERAD_RESET</p>
1-0	RESERVED	R	0h	Reserved

### 11.6.3.5 HWBP\_STATUS Register (Offset = 7h) [reset = 400h]

HWBP\_STATUS is shown in [Figure 11-12](#) and described in [Table 11-16](#).

Return to the [Summary Table](#).

HWBP Status Register

**Figure 11-12. HWBP\_STATUS Register**

15	14	13	12	11	10	9	8
STATUS				MODULE_ID			
R-0h				R-4h			
7	6	5	4	3	2	1	0
RESERVED							EVENT_FIRED
R-0h							R-0h

**Table 11-16. HWBP\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	STATUS	R	0h	Bus comparator status: 00 Idle 10 Enabled 11 Completed Reset type: ERAD_RESET
13-8	MODULE_ID	R	4h	These bits are always a constant representing a unique identification for the bus comparator unit. Reset type: ERAD_RESET
7-1	RESERVED	R	0h	Reserved
0	EVENT_FIRED	R	0h	This is a sticky bit which gets set every time the HWBP unit generates a match event. This will be used by software to figure out whether this HWBP module fired an event or not. This bit will get cleared by writing a '1' to bit 0 of the HWBP_CLEAR register. Reset type: ERAD_RESET

### 11.6.4 ERAD\_COUNTER\_REGS Registers

Table 11-17 lists the ERAD\_COUNTER\_REGS registers. All register offset addresses not listed in Table 11-17 should be considered as reserved locations and the register contents should not be modified.

**Table 11-17. ERAD\_COUNTER\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	CTM_CNTL	Counter Control Register	EALLOW	<a href="#">Go</a>
1h	CTM_STATUS	Counter Status Register	EALLOW	<a href="#">Go</a>
2h	CTM_REF	Counter Reference Register	EALLOW	<a href="#">Go</a>
4h	CTM_COUNT	Counter Current Value Register	EALLOW	<a href="#">Go</a>
6h	CTM_MAX_COUNT	Counter Max Count Value Register	EALLOW	<a href="#">Go</a>
8h	CTM_INPUT_SEL	Counter Input Select Register	EALLOW	<a href="#">Go</a>
9h	CTM_CLEAR	Counter Clear Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 11-18 shows the codes that are used for access types in this section.

**Table 11-18. ERAD\_COUNTER\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 11.6.4.1 CTM\_CNTL Register (Offset = 0h) [reset = 0h]

CTM\_CNTL is shown in [Figure 11-13](#) and described in [Table 11-19](#).

Return to the [Summary Table](#).

Counter Control Register

**Figure 11-13. CTM\_CNTL Register**

15	14	13	12	11	10	9	8
RST_INP_SEL					RST_EN	RESERVED	RESERVED
R/W-0h					R/W-0h	R-0h	
7	6	5	4	3	2	1	0
RTOSINT	STOP	RESERVED	RST_ON_MAT CH	EVENT_MODE	START_STOP_ MODE	RESERVED	
R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	

**Table 11-19. CTM\_CNTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RST_INP_SEL	R/W	0h	These 5 bits are used to select the event input that will be used as the reset input. These bits matter only if the Enable Reset bit is set to 1. Reset type: ERAD_RESET
10	RST_EN	R/W	0h	This bit decides if the reset input is enabled or not. Setting this to 1 will cause the counter to reset to zero whenever the selected reset input goes active high. No event will be generated when the counter is reset. Setting this bit to 0 will cause the counter to ignore the reset inputs. Reset type: ERAD_RESET
9	RESERVED	R	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RTOSINT	R/W	0h	This bit decides whether the counter module will generate RTOSINTn interrupt when count value matches the reference. Note that the event outputs will always be generated regardless of the state of this bit. 0 The counter unit will not cause any action towards the CPU. 1 The counter unit will assert RTOSINTn when the count value matches the reference value. Reset type: ERAD_RESET
6	STOP	R/W	0h	This bit decides whether the counter module will generate a watchpoint to the CPU when the count value matches the reference. Note that the event outputs will always be generated regardless of the state of this bit. 0 The counter unit will not generate a watchpoint. 1 The counter unit will assert ANASTOP when the count value matches the reference. Reset type: ERAD_RESET
5	RESERVED	R	0h	Reserved
4	RST_ON_MATCH	R/W	0h	This bit is used to decide whether the counter will reset to zero once it reaches the reference value. 0 Counter will stay at the reference value and the counter will go to COMPLETED state and further counting will be stopped. 1 The counter will reset to zero once it reaches the match value and will stay enabled. Reset type: ERAD_RESET



**Table 11-19. CTM\_CNTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	EVENT_MODE	R/W	0h	<p>This bit is used to decide whether the counter will count the level of the event or the edge of the event.</p> <p>0 Counter will increment the count as long as the count input is active high.</p> <p>1 The counter will count only on the rising edge of the count input. Reset type: ERAD_RESET</p>
2	START_STOP_MODE	R/W	0h	<p>This bit is used to decide whether the counter will count in the START_STOP mode or not.</p> <p>0 Normal count mode. The counter will not depend on the START and STOP events</p> <p>1 This is the START-STOP mode of the counter. The counter will start counting only after the START input has been asserted. It will continue to count the selected event till the STOP event is seen. Reset type: ERAD_RESET</p>
1-0	RESERVED	R	0h	Reserved

### 11.6.4.2 CTM\_STATUS Register (Offset = 1h) [reset = 10h]

CTM\_STATUS is shown in [Figure 11-14](#) and described in [Table 11-20](#).

Return to the [Summary Table](#).

Counter Status Register

**Figure 11-14. CTM\_STATUS Register**

15	14	13	12	11	10	9	8
STATUS				MODULE_ID			
R-0h				R-4h			
7	6	5	4	3	2	1	0
MODULE_ID						OVERFLOW	EVENT_FIRED
R-4h						R-0h	R-0h

**Table 11-20. CTM\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	STATUS	R	0h	Counter unit status,  00 Idle 10 Enabled 11 Completed Reset type: ERAD_RESET
11-2	MODULE_ID	R	4h	These bits are always a constant representing a unique identification for the trigger unit. Reset type: ERAD_RESET
1	OVERFLOW	R	0h	This is a sticky bit which gets set every time the counter overflows and wraps around after reaching 0xffffffff. This bit will get cleared by writing a '1' to bit 9 of the CTM_CNTL register. Reset type: ERAD_RESET
0	EVENT_FIRED	R	0h	This is a sticky bit which gets set every time the CTM unit generates a match event. This will be used by software to figure out whether this CTM module fired an event or not. This bit will get cleared by writing a '1' to bit 9 of the CTM_CNTL register. Reset type: ERAD_RESET

### 11.6.4.3 CTM\_REF Register (Offset = 2h) [reset = 0h]

CTM\_REF is shown in [Figure 11-15](#) and described in [Table 11-21](#).

Return to the [Summary Table](#).

Counter Reference Register

**Figure 11-15. CTM\_REF 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
REF																															
R/W-0h																															

**Table 11-21. CTM\_REF Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	REF	R/W	0h	<p>This register contains the counter reference value for comparison. The counter will generate an event if the count value matches the reference register (considering both upper and lower half of the register).</p> <p>This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored. The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored.</p> <p>Reference match is enabled only when a non zero value is programmed on one of the REF register.</p> <p>Reset type: ERAD_RESET</p>

#### 11.6.4.4 CTM\_COUNT Register (Offset = 4h) [reset = 0h]

CTM\_COUNT is shown in [Figure 11-16](#) and described in [Table 11-22](#).

Return to the [Summary Table](#).

Counter Current Value Register

**Figure 11-16. CTM\_COUNT 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
COUNT																															
R/W-0h																															

**Table 11-22. CTM\_COUNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	COUNT	R/W	0h	This register contains the current count value. The counter will generate an event if the count value matches the reference register (considering both upper and lower half of the register). This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored. The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored. Reset type: ERAD_RESET

### 11.6.4.5 CTM\_MAX\_COUNT Register (Offset = 6h) [reset = 0h]

CTM\_MAX\_COUNT is shown in [Figure 11-17](#) and described in [Table 11-23](#).

Return to the [Summary Table](#).

Counter Max Count Value Register

**Figure 11-17. CTM\_MAX\_COUNT 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
MAX_COUNT																															
R/W-0h																															

**Table 11-23. CTM\_MAX\_COUNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	MAX_COUNT	R/W	0h	<p>This register contains the maximum recorded counter value. This is relevant only in the Start Stop mode of operation.</p> <p>This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored.</p> <p>The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored.</p> <p>Reset type: ERAD_RESET</p>

**11.6.4.6 CTM\_INPUT\_SEL Register (Offset = 8h) [reset = 0h]**

CTM\_INPUT\_SEL is shown in [Figure 11-18](#) and described in [Table 11-24](#).

Return to the [Summary Table](#).

Counter Input Select Register

**Figure 11-18. CTM\_INPUT\_SEL Register**

15	14	13	12	11	10	9	8
STO_INP_SEL					STA_INP_SEL		
R/W-0h					R/W-0h		
7	6	5	4	3	2	1	0
STA_INP_SEL		CNT_INP_SEL				CTM_INP_SEL_EN	
R/W-0h		R/W-0h				R/W-0h	

**Table 11-24. CTM\_INPUT\_SEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	STO_INP_SEL	R/W	0h	These 5 bits decide which of the 32 inputs will be selected as the STOP event for the counter. The 32 inputs will be hooked up to the event outputs from the breakpoint module and to other system events. The usage of these bits are relevant only in the START_STOP mode of counting. Reset type: ERAD_RESET
10-6	STA_INP_SEL	R/W	0h	These 5 bits decide which of the 32 inputs will be selected as the START event for the counter. The 32 inputs will be hooked up to the event outputs from the breakpoint module and to other system events. The usage of these bits are relevant only in the START_STOP mode of counting. Reset type: ERAD_RESET
5-1	CNT_INP_SEL	R/W	0h	These 5 bits decide which of the 32 inputs will be selected to enable counting. The 32 inputs will be hooked up to the event outputs from the breakpoint module and to other system events. Reset type: ERAD_RESET
0	CTM_INP_SEL_EN	R/W	0h	Counter input select enable:  0 Disable using the input_select register for the count input. The counter will always count CPU cycles.  1 Enable using the input_select register for the count input. The counter will count the event selected by the count input register. Reset type: ERAD_RESET

**11.6.4.7 CTM\_CLEAR Register (Offset = 9h) [reset = 0h]**

CTM\_CLEAR is shown in [Figure 11-19](#) and described in [Table 11-25](#).

Return to the [Summary Table](#).

Counter Clear Register

**Figure 11-19. CTM\_CLEAR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						OVERFLOW_C LEAR	EVENT_CLEA R
R-0h						R-0/W-0h	R-0/W-0h

**Table 11-25. CTM\_CLEAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1	OVERFLOW_CLEAR	R-0/W	0h	Clear OVERFLOW:  0 No action. 1 A write with this bit set to 1 will clear the sticky OVERFLOW bit in the CTM_STATUS register. Reads of this bit position will always return a 0. Reset type: ERAD_RESET
0	EVENT_CLEAR	R-0/W	0h	Clear EVENT_FIRED:  0 No action. 1 A write with this bit set to 1 will clear the sticky EVENT_FIRED bit in the CTM_STATUS register and bring the Breakpoint Module statemachine status back to IDLE. Reads of this bit position will always return a 0. Reset type: ERAD_RESET

## Analog Subsystem

---

---

---

This analog subsystem module is described in this chapter.

Topic	Page
12.1 Introduction .....	1449
12.2 Analog Subsystem Registers .....	1458



## 12.1 Introduction

The analog modules on this device include the Analog-to-Digital Converter (ADC), Programmable Gain Amplifier (PGA), Temperature Sensor, Buffered Digital-to-Analog Converter (DAC), and Comparator Subsystem (CMPSS).

### 12.1.1 Features

The analog subsystem has the following features:

- Flexible voltage references
  - The ADCs are referenced to VREFHlx and VSSA pins
    - VREFHlx pin voltage can be driven in externally or can be generated by an internal bandgap voltage reference
    - The internal voltage reference range can be selected to be 0V to 3.3V or 0V to 2.5V
  - The buffered DACs are referenced to VREFHlx and VSSA
    - Alternately, these DACs can be referenced to the VDAC pin and VSSA
  - The comparator DACs are referenced to VDDA and VSSA
    - Alternately, these DACs can be referenced to the VDAC pin and VSSA
- Flexible pin usage
  - Buffered DAC outputs, comparator subsystem inputs, PGA functions, and digital inputs are multiplexed with ADC inputs
  - Internal connection to  $V_{REFLO}$  on all ADCs for offset self-calibration

### 12.1.2 Block Diagram

The subsystem block diagrams show the connections between the different integrated analog modules to the device pins. These pins fall into two categories: analog module inputs/outputs and reference pins.

The analog pins are organized into analog groups around a PGA and CMPSS module. The block diagram shows which pins connect to each group, but does not show the specific connections from the pins to the ADC, DAC, CMPSS, or PGA modules.

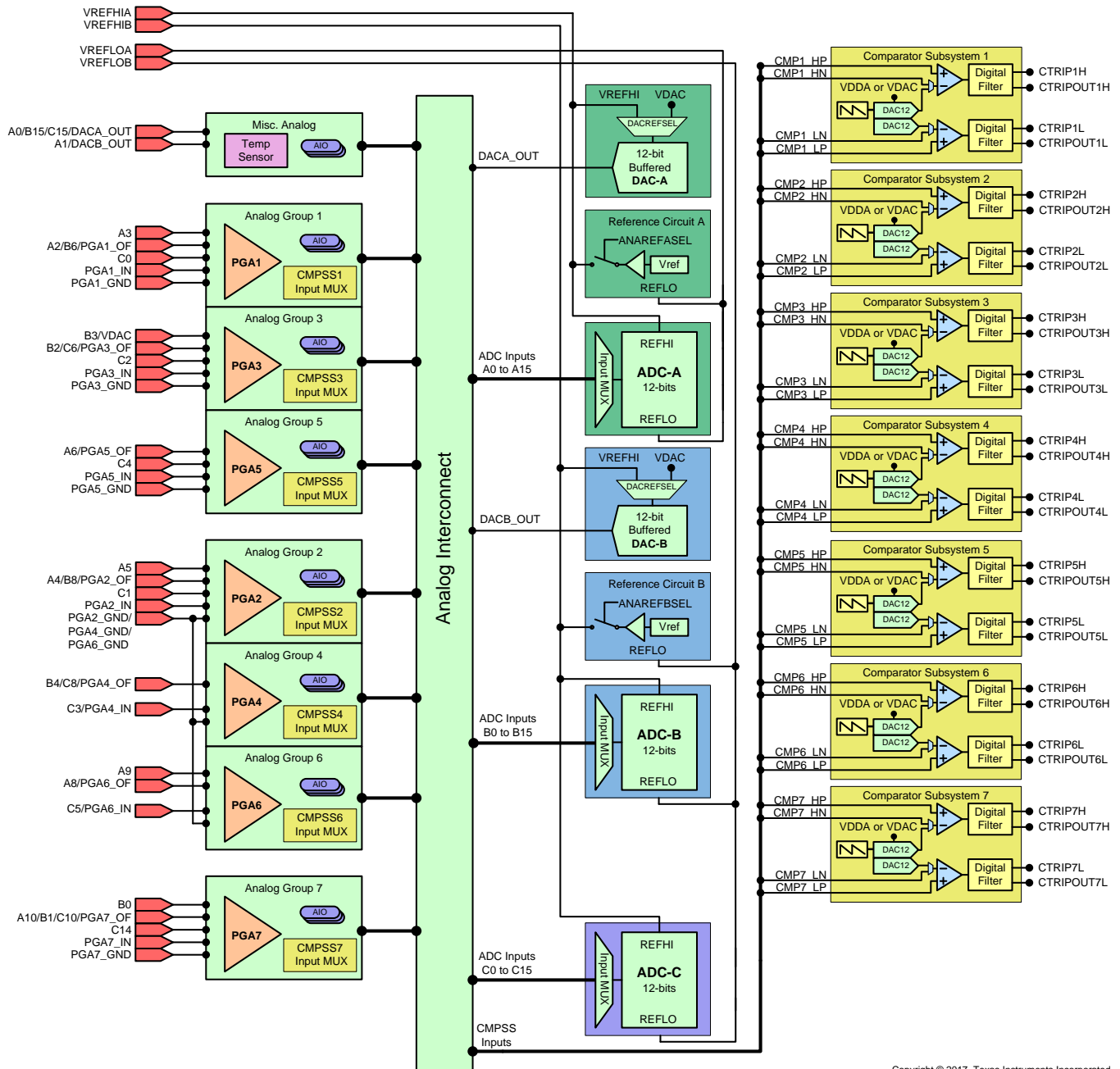
The block diagram for the analog subsystem is presented in the following graphics.

The following note applies to all packages.

#### NOTES:

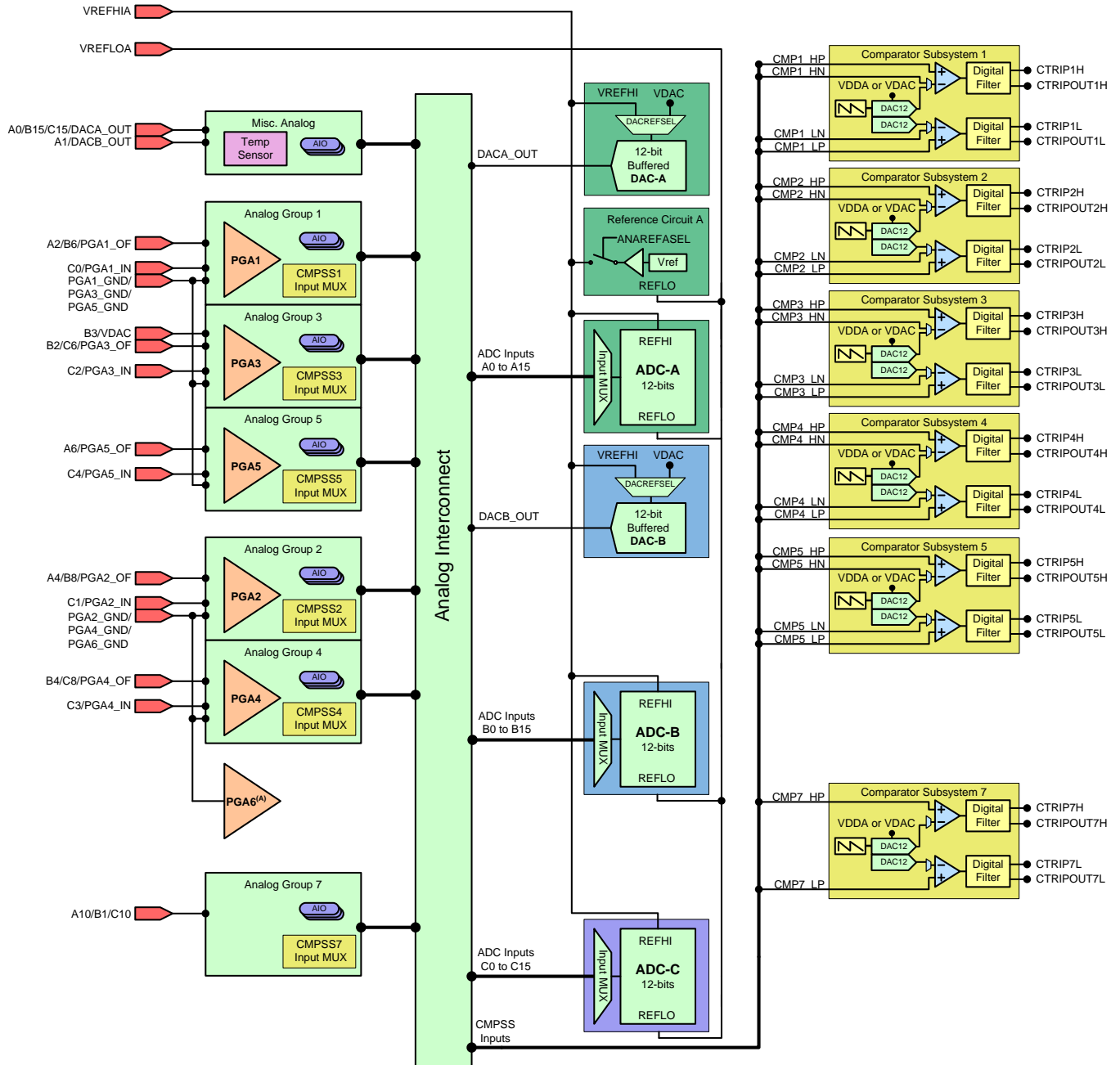
- Not all analog pins are available on all devices. Consult the datasheet for your specific device to determine which pins are available.
- Consult the datasheet for your device to determine the allowable voltage range for VREFHI and VREFLO
- An external capacitor is required on the VREFHI pins. Consult the datasheet for the specific value required.
- For buffered DAC modules, VSSA will be the low reference whether VREFHlx or VDAC is selected as the high reference.
- For CMPSS modules, VSSA will be the low reference whether VDAC or VDDA is selected as the high reference.

Figure 12-1. Analog Subsystem Block Diagram (100-Pin PZ LQFP)



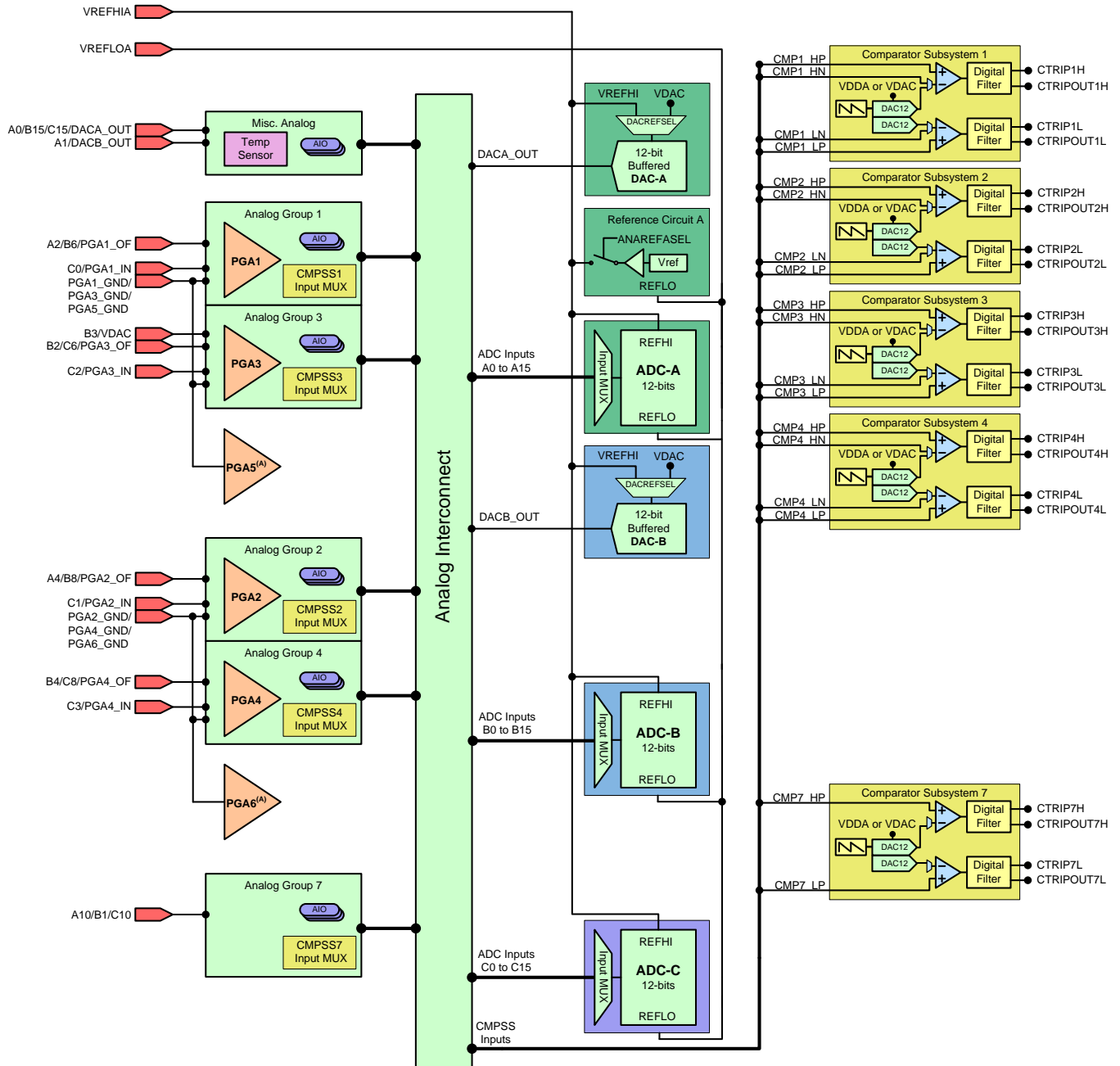
Copyright © 2017, Texas Instruments Incorporated

Figure 12-2. Analog Subsystem Block Diagram (64-Pin PM LQFP)



Copyright © 2017, Texas Instruments Incorporated

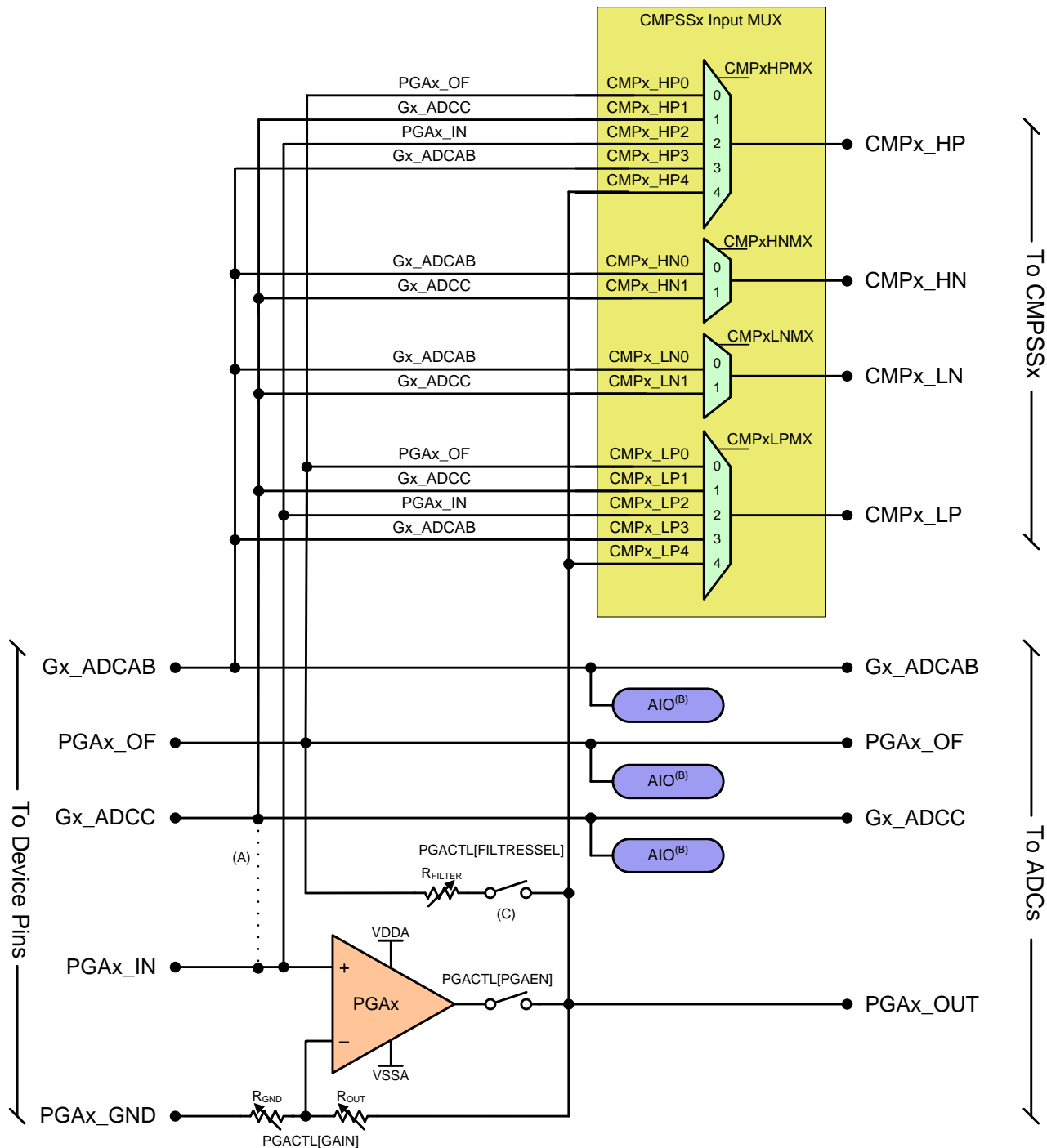
Figure 12-3. Analog Subsystem Block Diagram (56-Pin RSH VQFN)



Copyright © 2017, Texas Instruments Incorporated

Figure 12-4 shows how each analog group is structured.

Figure 12-4. Analog Group Connections



- A On lower pin-count packages, the input to ADCC will share a pin with the PGA input. If the PGA input is unused, then the ADCC input can allow the pin to be used as an ADC input, a negative comparator input, or a digital input.
- B AIOs support digital input mode only.

Table 12-1. Analog Pins and Internal Connections

PIN NAME	GROUP NAME	PACKAGE			ALWAYS CONNECTED (NO MUX)					COMPARATOR SUBSYSTEM (MUX)				AIO INPUT	
		100 PZ	64 PM	56 RSH	ADCA	ADCB	ADCC	PGA	DAC	HIGH POSITIVE	HIGH NEGATIVE	LOW POSITIVE	LOW NEGATIVE		
VREFHIA	-	25	16	14											
VREFHIB	-	24													
VREFHIC	-														
VREFLOA	-	27	17	15	A13										
VREFLOB	-	26				B13									
VREFLOC	-							C13							
<b>Analog Group 1</b>										<b>CMP1</b>					
A3	G1_ADCAB	10			A3					HPMXSEL = 3	HNMXSEL = 0	LPMXSEL = 3	LNMXSEL = 0	AIO233	
A2/B6/PGA1_OF	PGA1_OF	9	9	8	A2	B6		PGA1_OF		HPMXSEL = 0		LPMXSEL = 0		AIO224	
C0	G1_ADCC	19	12	10			C0			HPMXSEL = 1	HNMXSEL = 1	LPMXSEL = 1	LNMXSEL = 1	AIO237	
PGA1_IN	PGA1_IN	18							PGA1_IN		HPMXSEL = 2		LPMXSEL = 2		
PGA1_GND	PGA1_GND	14	10	9				PGA1_GND							
-	PGA1_OUT <sup>(1)</sup>				A11	B7		PGA1_OUT		HPMXSEL = 4		LPMXSEL = 4			
<b>Analog Group 2</b>										<b>CMP2</b>					
A5	G2_ADCAB	35			A5					HPMXSEL = 3	HNMXSEL = 0	LPMXSEL = 3	LNMXSEL = 0	AIO234	
A4/B8/PGA2_OF	PGA2_OF	36	23	21	A4	B8		PGA2_OF		HPMXSEL = 0		LPMXSEL = 0		AIO225	
C1	G2_ADCC	29	18	16			C1			HPMXSEL = 1	HNMXSEL = 1	LPMXSEL = 1	LNMXSEL = 1	AIO238	
PGA2_IN	PGA2_IN	30							PGA2_IN		HPMXSEL = 2		LPMXSEL = 2		
PGA2_GND	PGA2_GND	32	20	18				PGA2_GND							
-	PGA2_OUT <sup>(1)</sup>				A12	B9		PGA2_OUT		HPMXSEL = 4		LPMXSEL = 4			
<b>Analog Group 3</b>										<b>CMP3</b>					
B3/VDAC	G3_ADCAB	8	8	7		B3			VDAC	HPMXSEL = 3	HNMXSEL = 0	LPMXSEL = 3	LNMXSEL = 0	AIO242	
B2/C6/PGA3_OF	PGA3_OF	7	7	6		B2	C6	PGA3_OF		HPMXSEL = 0		LPMXSEL = 0		AIO226	
C2	G3_ADCC	21	13	11			C2			HPMXSEL = 1	HNMXSEL = 1	LPMXSEL = 1	LNMXSEL = 1	AIO244	
PGA3_IN	PGA3_IN	20							PGA3_IN		HPMXSEL = 2		LPMXSEL = 2		
PGA3_GND	PGA3_GND	15	10	9				PGA3_GND							
-	PGA3_OUT <sup>(1)</sup>					B10	C7	PGA3_OUT		HPMXSEL = 4		LPMXSEL = 4			
<b>Analog Group 4</b>										<b>CMP4</b>					
B5	G4_ADCAB					B5				HPMXSEL = 3	HNMXSEL = 0	LPMXSEL = 3	LNMXSEL = 0	AIO243	
B4/C8/PGA4_OF	PGA4_OF	39	24	22		B4	C8	PGA4_OF		HPMXSEL = 0		LPMXSEL = 0		AIO227	
C3	G4_ADCC	31	19	17			C3			HPMXSEL = 1	HNMXSEL = 1	LPMXSEL = 1	LNMXSEL = 1	AIO245	
PGA4_IN	PGA4_IN									PGA4_IN		HPMXSEL = 2		LPMXSEL = 2	
PGA4_GND	PGA4_GND	32	20	18				PGA4_GND							
-	PGA4_OUT <sup>(1)</sup>					B11	C9	PGA4_OUT		HPMXSEL = 4		LPMXSEL = 4			

<sup>(1)</sup> Internal connection only; does not come to a device pin.

**Table 12-1. Analog Pins and Internal Connections (continued)**

PIN NAME	GROUP NAME	PACKAGE			ALWAYS CONNECTED (NO MUX)					COMPARATOR SUBSYSTEM (MUX)				AIO INPUT	
		100 PZ	64 PM	56 RSH	ADCA	ADCB	ADCC	PGA	DAC	HIGH POSITIVE	HIGH NEGATIVE	LOW POSITIVE	LOW NEGATIVE		
<b>Analog Group 5</b>											<b>CMP5</b>				
A7	G5_ADCAB				A7					HPMXSEL = 3	HNMXSEL = 0	LPMXSEL = 3	LNMXSEL = 0	AIO235	
A6/PGA5_OF	PGA5_OF	6	6		A6			PGA5_OF		HPMXSEL = 0		LPMXSEL = 0		AIO228	
C4	G5_ADCC	17	11				C4			HPMXSEL = 1	HNMXSEL = 1	LPMXSEL = 1	LNMXSEL = 1	AIO239	
PGA5_IN	PGA5_IN	16						PGA5_IN		HPMXSEL = 2		LPMXSEL = 2			
PGA5_GND	PGA5_GND	13	10	9				PGA5_GND							
-	PGA5_OUT <sup>(1)</sup>				A14			PGA5_OUT		HPMXSEL = 4		LPMXSEL = 4			
<b>Analog Group 6</b>											<b>CMP6</b>				
A9	G6_ADCAB	38			A9					HPMXSEL = 3	HNMXSEL = 0	LPMXSEL = 3	LNMXSEL = 0	AIO236	
A8/PGA6_OF	PGA6_OF	37			A8			PGA6_OF		HPMXSEL = 0		LPMXSEL = 0		AIO229	
C5	G6_ADCC	28					C5			HPMXSEL = 1	HNMXSEL = 1	LPMXSEL = 1	LNMXSEL = 1	AIO240	
PGA6_IN	PGA6_IN							PGA6_IN		HPMXSEL = 2		LPMXSEL = 2			
PGA6_GND	PGA6_GND	32	20	18				PGA6_GND							
-	PGA6_OUT <sup>(1)</sup>				A15			PGA6_OUT		HPMXSEL = 4		LPMXSEL = 4			
<b>Analog Group 7</b>											<b>CMP7</b>				
B0	G7_ADCAB	41				B0				HPMXSEL = 3	HNMXSEL = 0	LPMXSEL = 3	LNMXSEL = 0	AIO241	
A10/B1/C10/PGA7_OF	PGA7_OF <sup>(2)</sup>	40	25	23	A10	B1	C10	PGA7_OF		HPMXSEL = 0		LPMXSEL = 0		AIO230	
C14	G7_ADCC	44					C14			HPMXSEL = 1	HNMXSEL = 1	LPMXSEL = 1	LNMXSEL = 1	AIO246	
PGA7_IN	PGA7_IN	43						PGA7_IN		HPMXSEL = 2		LPMXSEL = 2			
PGA7_GND	PGA7_GND	42						PGA7_GND							
-	PGA7_OUT <sup>(1)</sup>					B12	C11	PGA7_OUT		HPMXSEL = 4		LPMXSEL = 4			
<b>Other Analog</b>															
A0/B15/C15/DACA_OUT		23	15	13	A0	B15	C15		DACA_OUT					AIO231	
A1/DACB_OUT		22	14	12	A1				DACB_OUT					AIO232	
C12							C12							AIO247	
-	TempSensor <sup>(1)</sup>					B14									

<sup>(2)</sup> PGA functionality not available on 64-pin and 56-pin packages.

**Table 12-2. Analog Signal Descriptions**

SIGNAL NAME	DESCRIPTION
AIOx	Digital input on ADC pin
Ax	ADC A Input
Bx	ADC B Input
Cx	ADC C Input
CMPx_HNy	Comparator subsystem high comparator negative input
CMPx_HPy	Comparator subsystem high comparator positive input
CMPx_LNy	Comparator subsystem low comparator negative input
CMPx_LPy	Comparator subsystem low comparator positive input
DACx_OUT	Buffered DAC Output
PGAx_GND	PGA Ground
PGAx_IN	PGA Input
PGAx_OF	PGA Output for filter
PGAx_OUT	PGA Output to internal ADC
TempSensor	Internal temperature sensor
VDAC	Optional external reference voltage for on-chip DACs. There is a 100-pF capacitor to VSSA on this pin whether used for ADC input or DAC reference which cannot be disabled. If this pin is being used as a reference for the on-chip DACs, place at least a 1-uF capacitor on this pin.

**Table 12-3. Reference Summary**

Module	Reference Option	Configured Where?	Register	Driverlib Function	Notes
ADC	Internal or External	Analog System	AnalogSubsysRegs. ANAREFCTL.bit.ANAREFxSEL	ADC_setVREF	Both options require use of the VREFHI pin
	3.3V or 2.5V Internal Reference Range	Analog System	AnalogSubsysRegs. ANAREFCTL.bit.ANAREFx2P5SEL	ADC_setVREF	Only applicable when using internal reference mode.
Buffered DAC	VREFHI or VDAC	DAC Module	DacxRegs. DACCTL.bit.DACREFSSEL	DAC_setReferenceVoltage	
	Internal or External	Analog System	(follows ADC configuration)	ADC_setVREF	Only applicable when using VREFHI
	3.3V or 2.5V Internal Reference Range	Analog System	(follows ADC configuration)	ADC_setVREF	Only applicable when using VREFHI with internal reference
		DAC Module	DacxRegs. DACCTL.bit.MODE	DAC_setGainMode	Gain = 2 should be set when using 3.3V internal reference mode.
CMPSS DACs	VDDA or VDAC	CMPSS Module	CmpssxRegs. COMPDACCTL.bit.SELREF	CMPSS_configDAC	



### **12.1.3 Lock Registers**

Setting one of the bits in the lock registers prevents further writes to the associated analog subsystem configuration register. This lock can only be cleared by a device reset.

## 12.2 Analog Subsystem Registers

This section describes the Analog Subsystem Registers.

### 12.2.1 Analog Subsystem Base Addresses

**Table 12-4. Analog Subsystem Base Address Table**

Device Registers	Register Name	Start Address	End Address
AnalogSubsysRegs	ANALOG_SUBSYS_REGS	0x0005_D700	0x0005_D7FF

## 12.2.2 ANALOG\_SUBSYS\_REGS Registers

Table 12-5 lists the ANALOG\_SUBSYS\_REGS registers. All register offset addresses not listed in Table 12-5 should be considered as reserved locations and the register contents should not be modified.

**Table 12-5. ANALOG\_SUBSYS\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
1Eh	ANAREFPP	ADC Analog Reference Peripheral Properties register. The value of this register is populated during boot rom.	EALLOW	<a href="#">Go</a>
60h	TSNSCTL	Temperature Sensor Control Register	EALLOW	<a href="#">Go</a>
68h	ANAREFCTL	Analog Reference Control Register	EALLOW	<a href="#">Go</a>
70h	VMONCTL	Voltage Monitor Control Register	EALLOW	<a href="#">Go</a>
78h	DCDCCTL	DC-DC control register.	EALLOW	<a href="#">Go</a>
7Ah	DCDCSTS	DC-DC status register.		<a href="#">Go</a>
82h	CMPPMXSEL	Bits to select one of the many sources on CopmHP inputs. Refer to Pimux diagram for details.	EALLOW	<a href="#">Go</a>
84h	CMPLPMXSEL	Bits to select one of the many sources on CopmLP inputs. Refer to Pimux diagram for details.	EALLOW	<a href="#">Go</a>
86h	CMPHNMXSEL	Bits to select one of the many sources on CopmHN inputs. Refer to Pimux diagram for details.	EALLOW	<a href="#">Go</a>
87h	CMPLNMXSEL	Bits to select one of the many sources on CopmLN inputs. Refer to Pimux diagram for details.	EALLOW	<a href="#">Go</a>
8Eh	LOCK	Lock Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 12-6 shows the codes that are used for access types in this section.

**Table 12-6. ANALOG\_SUBSYS\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
WOnce	WOnce	Write Write once
WSonce	WSonce	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		

**Table 12-6. ANALOG\_SUBSYS\_REGS Access Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 12.2.2.1 ANAREFPP Register (Offset = 1Eh) [reset = 0h]

ANAREFPP is shown in [Figure 12-5](#) and described in [Table 12-7](#).

Return to the [Summary Table](#).

ADC Analog Reference Peripheral Properties register. The value of this register is populated during boot rom.

**Figure 12-5. ANAREFPP Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						ANAREFCDIS	ANAREFBDIS
R-0h						R/WOnce-0h	R/WOnce-0h

**Table 12-7. ANAREFPP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1	ANAREFCDIS	R/WOnce	0h	ANAREFC Disable. This bit field determines, whether ANAREFC is disabled or enabled. 0 ANAREFC is enabled. 1 ANAREFC is disabled.  Note: This bit should be programmed to 1 in parts where VREFHIA and VREFHIB are double bonded.  Reset type: XRSn
0	ANAREFBDIS	R/WOnce	0h	ANAREFB Disable. This bit field determines, whether ANAREFB is disabled or enabled. 0 ANAREFB is enabled. 1 ANAREFB is disabled.  Note: This bit should be programmed to 1 in parts where VREFHIA, VREFHIB and VREFHIC are triple bonded.  Reset type: XRSn

**12.2.2.2 TSNSCTL Register (Offset = 60h) [reset = 0h]**

TSNSCTL is shown in [Figure 12-6](#) and described in [Table 12-8](#).

Return to the [Summary Table](#).

Temperature Sensor Control Register

**Figure 12-6. TSNSCTL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R-0h							R/W-0h

**Table 12-8. TSNSCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	ENABLE	R/W	0h	Temperature Sensor Enable. This bit enables the temperature sensor output to the ADC.  0 Disabled 1 Enabled Reset type: SYSRSn

### 12.2.2.3 ANAREFCTL Register (Offset = 68h) [reset = Fh]

ANAREFCTL is shown in [Figure 12-7](#) and described in [Table 12-9](#).

Return to the [Summary Table](#).

Analog Reference Control Register

**Figure 12-7. ANAREFCTL Register**

15	14	13	12	11	10	9	8
RESERVED					ANAREFC2P5SEL	ANAREFB2P5SEL	ANAREFA2P5SEL
R-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED					ANAREFCSEL	ANAREFBSEL	ANAREFASEL
R-1h					R/W-1h	R/W-1h	R/W-1h

**Table 12-9. ANAREFCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved
10	ANAREFC2P5SEL	R/W	0h	<p>Analog referenc C 2.5V source select. In internal reference mode, this bit selects which voltage the internal reference buffer drives onto the VREFHI pin. The buffer can drive either 1.65V onto the pin, resulting in a reference range of 0 to 3.3V, or the buffer can drive 2.5V onto the pin, resulting in a reference range of 0 to 2.5V. If switching between these two modes, the user must allow adequate time for the external capacitor to charge to the new voltage before using the ADC or buffered DAC. If multiple VREFHI pins are ganged together (for lower pin-count packages), then the reference voltage select for the ganged pins should always be configured to the same setting.</p> <p>0 Internal 1.65V reference mode (3.3V reference range) 1 Internal 2.5V reference mode (2.5V reference range)</p> <p>Reset type: XRSn</p>
9	ANAREFB2P5SEL	R/W	0h	<p>Analog referenc B 2.5V source select. In internal reference mode, this bit selects which voltage the internal reference buffer drives onto the VREFHI pin. The buffer can drive either 1.65V onto the pin, resulting in a reference range of 0 to 3.3V, or the buffer can drive 2.5V onto the pin, resulting in a reference range of 0 to 2.5V. If switching between these two modes, the user must allow adequate time for the external capacitor to charge to the new voltage before using the ADC or buffered DAC. If multiple VREFHI pins are ganged together (for lower pin-count packages), then the reference voltage select for the ganged pins should always be configured to the same setting.</p> <p>0 Internal 1.65V reference mode (3.3V reference range) 1 Internal 2.5V reference mode (2.5V reference range)</p> <p>Reset type: XRSn</p>

**Table 12-9. ANAREFCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	ANAREFA2P5SEL	R/W	0h	<p>Analog referenc A 2.5V source select. In internal reference mode, this bit selects which voltage the internal reference buffer drives onto the VREFHI pin. The buffer can drive either 1.65V onto the pin, resulting in a reference range of 0 to 3.3V, or the buffer can drive 2.5V onto the pin, resulting in a reference range of 0 to 2.5V. If switching between these two modes, the user must allow adequate time for the external capacitor to charge to the new voltage before using the ADC or buffered DAC. If multiple VREFHI pins are ganged together (for lower pin-count packages), then the reference voltage select for the ganged pins should always be configured to the same setting.</p> <p>0 Internal 1.65V reference mode (3.3V reference range) 1 Internal 2.5V reference mode (2.5V reference range)</p> <p>Reset type: XRSn</p>
7-3	RESERVED	R	1h	Reserved
2	ANAREFCSEL	R/W	1h	<p>Analog reference C mode select. This bit selects whether the VREFHIC pin uses internal reference mode (the device drives a voltage onto the VREFHI pin) or external reference mode (the system is expected to drive a voltage into the VREFHI pin). If multiple VREFHI pins are ganged together (for lower pin-count packages), then the mode select for the ganged pins should always be configured to the same setting</p> <p>0 Internal reference mode 1 External reference mode</p> <p>Reset type: XRSn</p>
1	ANAREFBSEL	R/W	1h	<p>Analog reference B mode select. This bit selects whether the VREFHIB pin uses internal reference mode (the device drives a voltage onto the VREFHI pin) or external reference mode (the system is expected to drive a voltage into the VREFHI pin). If multiple VREFHI pins are ganged together (for lower pin-count packages), then the mode select for the ganged pins should always be configured to the same setting</p> <p>0 Internal reference mode 1 External reference mode</p> <p>Reset type: XRSn</p>
0	ANAREFASEL	R/W	1h	<p>Analog reference A mode select. This bit selects whether the VREFHIA pin uses internal reference mode (the device drives a voltage onto the VREFHI pin) or external reference mode (the system is expected to drive a voltage into the VREFHI pin). If multiple VREFHI pins are ganged together (for lower pin-count packages), then the mode select for the ganged pins should always be configured to the same setting</p> <p>0 Internal reference mode 1 External reference mode</p> <p>Reset type: XRSn</p>



**12.2.2.4 VMONCTL Register (Offset = 70h) [reset = 0h]**

VMONCTL is shown in [Figure 12-8](#) and described in [Table 12-10](#).

Return to the [Summary Table](#).

Voltage Monitor Control Register

**Figure 12-8. VMONCTL Register**

15	14	13	12	11	10	9	8
RESERVED							BORLVMONDIS
R-0h							R/W-0h
7	6	5	4	3	2	1	0
RESERVED						RESERVED	RESERVED
R-0h							

**Table 12-10. VMONCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	BORLVMONDIS	R/W	0h	BORL disable on VDDIO.  0 BORL is enabled on VDDIO, i.e BOR circuit will be triggered if VDDIO goes lower than the lower BOR threshold of VDDIO. 1 BORL is disabled on VDDIO, i.e BOR circuit will not be triggered if VDDIO goes lower than the lower BOR threshold of VDDIO. Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

**12.2.2.5 DCDCCTL Register (Offset = 78h) [reset = 8000000h]**

DCDCCTL is shown in [Figure 12-9](#) and described in [Table 12-11](#).

Return to the [Summary Table](#).

DC-DC control register.

**Figure 12-9. DCDCCTL Register**

31	30	29	28	27	26	25	24
RESERVED		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							DCDCEN
R-0h							R/W-0h

**Table 12-11. DCDCCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30-1	RESERVED	R	0h	Reserved
0	DCDCEN	R/W	0h	Enable DC-DC. 0 : Disables DC-DC and the device would work of internal VREG. 1 : Enables DC-DC. Reset type: XRSn

**12.2.2.6 DCDCSTS Register (Offset = 7Ah) [reset = 0h]**

DCDCSTS is shown in [Figure 12-10](#) and described in [Table 12-12](#).

Return to the [Summary Table](#).

DC-DC status register.

**Figure 12-10. DCDCSTS Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					RESERVED	SWSEQDONE	INDETECT
R-0h						R-0h	R/W1S-0h

**Table 12-12. DCDCSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	SWSEQDONE	R	0h	<p>DC-DC switch sequence done.</p> <p>0 : Indicates that the sequence to switch to DC-DC is not complete. 1 : Indicates that the sequence to switch to DC-DC is complete.</p> <p>When DCDCCTL.DCDCEN is set, PMM does the necessary sequencing to switch to DC-DC, and at the end of the sequence this bit will be set. However the power source will be switched to DC-DC only if inductor functionality check passes, else the device will continue to work of VREG.</p> <p>Reset type: XRSn</p>
0	INDETECT	R/W1S	0h	<p>Inductor Detected Status.</p> <p>1 : Indicates that the external inductor connected to DC-DC is functional. 0 : Indicates that the external inductor connected to DC-DC is faulty.</p> <p>When DCDCCTL.DCDCEN is set, PMM checks for proper functioning of the external inductor. If the check shows that inductor is functional then this bit will be set.</p> <p>This status of this bit should be checked after DCDCSTS.SWSEQDONE is set.</p> <p>Reset type: XRSn</p>

**12.2.2.7 CMPHPMXSEL Register (Offset = 82h) [reset = 0h]**

CMPHPMXSEL is shown in [Figure 12-11](#) and described in [Table 12-13](#).

Return to the [Summary Table](#).

Bits to select one of the many sources on CopmHP inputs. Refer to Pimux diagram for details.

**Figure 12-11. CMPHPMXSEL Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED		CMP7HPMXSEL			CMP6HPMXSEL		
R-0h		R/W-0h			R/W-0h		
15	14	13	12	11	10	9	8
RESERVED	CMP5HPMXSEL			CMP4HPMXSEL			CMP3HPMXSEL
R-0h	R/W-0h			R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
CMP3HPMXSEL		CMP2HPMXSEL			CMP1HPMXSEL		
R/W-0h		R/W-0h			R/W-0h		

**Table 12-13. CMPHPMXSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0h	Reserved
21-19	CMP7HPMXSEL	R/W	0h	CMP-4HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
18-16	CMP6HPMXSEL	R/W	0h	CMP-5HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
15	RESERVED	R	0h	Reserved
14-12	CMP5HPMXSEL	R/W	0h	CMP5HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
11-9	CMP4HPMXSEL	R/W	0h	CMP4HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
8-6	CMP3HPMXSEL	R/W	0h	CMP3HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn

**Table 12-13. CMPHPMXSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-3	CMP2HPMXSEL	R/W	0h	<p>CMP2HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.</p> <p>Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn</p>
2-0	CMP1HPMXSEL	R/W	0h	<p>CMP1HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.</p> <p>Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn</p>

**12.2.2.8 CMPLPMXSEL Register (Offset = 84h) [reset = 0h]**

CMPLPMXSEL is shown in [Figure 12-12](#) and described in [Table 12-14](#).

Return to the [Summary Table](#).

Bits to select one of the many sources on CopmLP inputs. Refer to Pimux diagram for details.

**Figure 12-12. CMPLPMXSEL Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED		CMP7LPMXSEL			CMP6LPMXSEL		
R-0h		R/W-0h			R/W-0h		
15	14	13	12	11	10	9	8
RESERVED	CMP5LPMXSEL			CMP4LPMXSEL		CMP3LPMXSEL	
R-0h	R/W-0h			R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
CMP3LPMXSEL		CMP2LPMXSEL			CMP1LPMXSEL		
R/W-0h		R/W-0h			R/W-0h		

**Table 12-14. CMPLPMXSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0h	Reserved
21-19	CMP7LPMXSEL	R/W	0h	CMP7LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
18-16	CMP6LPMXSEL	R/W	0h	CMP6LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
15	RESERVED	R	0h	Reserved
14-12	CMP5LPMXSEL	R/W	0h	CMP5LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
11-9	CMP4LPMXSEL	R/W	0h	CMP4LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
8-6	CMP3LPMXSEL	R/W	0h	CMP3LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn

**Table 12-14. CMPLPMXSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-3	CMP2LPMXSEL	R/W	0h	<p>CMP2LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.</p> <p>Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn</p>
2-0	CMP1LPMXSEL	R/W	0h	<p>CMP1LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.</p> <p>Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn</p>

### 12.2.2.9 CMPHNMXSEL Register (Offset = 86h) [reset = 0h]

CMPHNMXSEL is shown in [Figure 12-13](#) and described in [Table 12-15](#).

Return to the [Summary Table](#).

Bits to select one of the many sources on CopmHN inputs. Refer to Pimux diagram for details.

**Figure 12-13. CMPHNMXSEL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	CMP7HNMXSEL L	CMP6HNMXSEL L	CMP5HNMXSEL L	CMP4HNMXSEL L	CMP3HNMXSEL L	CMP2HNMXSEL L	CMP1HNMXSEL L
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 12-15. CMPHNMXSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	CMP7HNMXSEL	R/W	0h	CMP7HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
5	CMP6HNMXSEL	R/W	0h	CMP6HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
4	CMP5HNMXSEL	R/W	0h	CMP5HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
3	CMP4HNMXSEL	R/W	0h	CMP4HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
2	CMP3HNMXSEL	R/W	0h	CMP3HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
1	CMP2HNMXSEL	R/W	0h	CMP2HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
0	CMP1HNMXSEL	R/W	0h	CMP1HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn



### 12.2.2.10 CMPLNMXSEL Register (Offset = 87h) [reset = 0h]

CMPLNMXSEL is shown in [Figure 12-14](#) and described in [Table 12-16](#).

Return to the [Summary Table](#).

Bits to select one of the many sources on CopmLN inputs. Refer to Pimux diagram for details.

**Figure 12-14. CMPLNMXSEL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	CMP7LNMXSE L	CMP6LNMXSE L	CMP5LNMXSE L	CMP4LNMXSE L	CMP3LNMXSE L	CMP2LNMXSE L	CMP1LNMXSE L
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 12-16. CMPLNMXSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	CMP7LNMXSEL	R/W	0h	CMP7LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
5	CMP6LNMXSEL	R/W	0h	CMP6LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
4	CMP5LNMXSEL	R/W	0h	CMP5LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
3	CMP4LNMXSEL	R/W	0h	CMP4LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
2	CMP3LNMXSEL	R/W	0h	CMP3LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
1	CMP2LNMXSEL	R/W	0h	CMP2LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn
0	CMP1LNMXSEL	R/W	0h	CMP1LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux. Reset type: XRSn

**12.2.2.11 LOCK Register (Offset = 8Eh) [reset = 0h]**

 LOCK is shown in [Figure 12-15](#) and described in [Table 12-17](#).

 Return to the [Summary Table](#).

Lock Register

**Figure 12-15. LOCK 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						VREGCTL	CMPLNMXSEL
R-0h						R/WOnce-0h	R/WOnce-0h
7	6	5	4	3	2	1	0
CMPHNMXSEL	CMPLPMXSEL	CMPPMXSEL	ADCINMXSEL	DCDCCTL	VMONCTL	ANAREFCTL	TSNSCTL
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 12-17. LOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	VREGCTL	R/WOnce	0h	VREGCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
8	CMPLNMXSEL	R/WOnce	0h	CMPLNMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
7	CMPHNMXSEL	R/WOnce	0h	CMPHNMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
6	CMPLPMXSEL	R/WOnce	0h	CMPLPMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
5	CMPPMXSEL	R/WOnce	0h	CMPPMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
4	ADCINMXSEL	R/WOnce	0h	ADCINMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
3	DCDCCTL	R/WOnce	0h	DCDCCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
2	VMONCTL	R/WOnce	0h	VMONCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn

**Table 12-17. LOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	ANAREFCTL	R/WOnce	0h	ANAREFCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
0	TSNSCTL	R/WOnce	0h	TSNSCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn

## Analog-to-Digital Converter (ADC)

The analog-to-digital converter module described in this chapter is a Type 5 ADC. See the *TMS320C28xx, 28xxx DSP Peripheral Reference Guide (SPRU566)* for a list of all devices with modules of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

Topic	Page
13.1 Introduction .....	1477
13.2 ADC Features .....	1477
13.3 ADC Block Diagram .....	1478
13.4 ADC Configurability .....	1478
13.5 SOC Principle of Operation .....	1481
13.6 SOC Configuration Examples .....	1484
13.7 ADC Conversion Priority.....	1486
13.8 Burst Mode .....	1489
13.9 EOC and Interrupt Operation .....	1491
13.10 Post-Processing Blocks .....	1492
13.11 Opens/Shorts Detection Circuit (OSDETECT).....	1496
13.12 Power-Up Sequence.....	1497
13.13 ADC Calibration.....	1498
13.14 ADC Timings.....	1498
13.15 Additional Information.....	1501
13.16 ADC Registers.....	1509

## 13.1 Introduction

The ADC module is a 12-bit successive approximation (SAR) style ADC. The ADC is composed of a core and a wrapper. The core is composed of the analog circuits which include the channel select MUX, the sample-and-hold (S/H) circuit, the successive approximation circuits, voltage reference circuits, and other analog support circuits. The wrapper is composed of the digital circuits that configure and control the ADC. These circuits include the logic for programmable conversions, result registers, interfaces to analog circuits, interfaces to the peripheral buses, post-processing circuits, and interfaces to other on-chip modules.

Each ADC module consists of a single sample-and-hold (s/h) circuit. The ADC module is designed to be duplicated multiple times on the same chip, allowing simultaneous sampling or independent operation of multiple ADCs. The ADC wrapper is start-of-conversion (SOC) based (see [Section 13.5](#)).

## 13.2 ADC Features

Each ADC has the following features:

- 12-bit resolution
- Ratiometric external reference set by VREFHI and VREFLO pins
- Selectable internal reference of 2.5V or 3.3V
- Single-ended signal conversions
- Input multiplexer with up to 16 channels
- 16 configurable SOCs
- 16 individually addressable result registers
- Multiple trigger sources
  - S/W - software immediate start
  - All ePWMs - ADCSOC A or B
  - GPIO XINT2
  - CPU Timers 0/1/2
  - ADCINT1/2
- Four flexible PIE interrupts
- Configurable interrupt placement
- Burst mode
- Four post-processing blocks, each with:
  - Saturating offset calibration
  - Error from setpoint calculation
  - High, low, and zero-crossing compare, with interrupt and ePWM trip capability
  - Trigger-to-sample delay capture

---

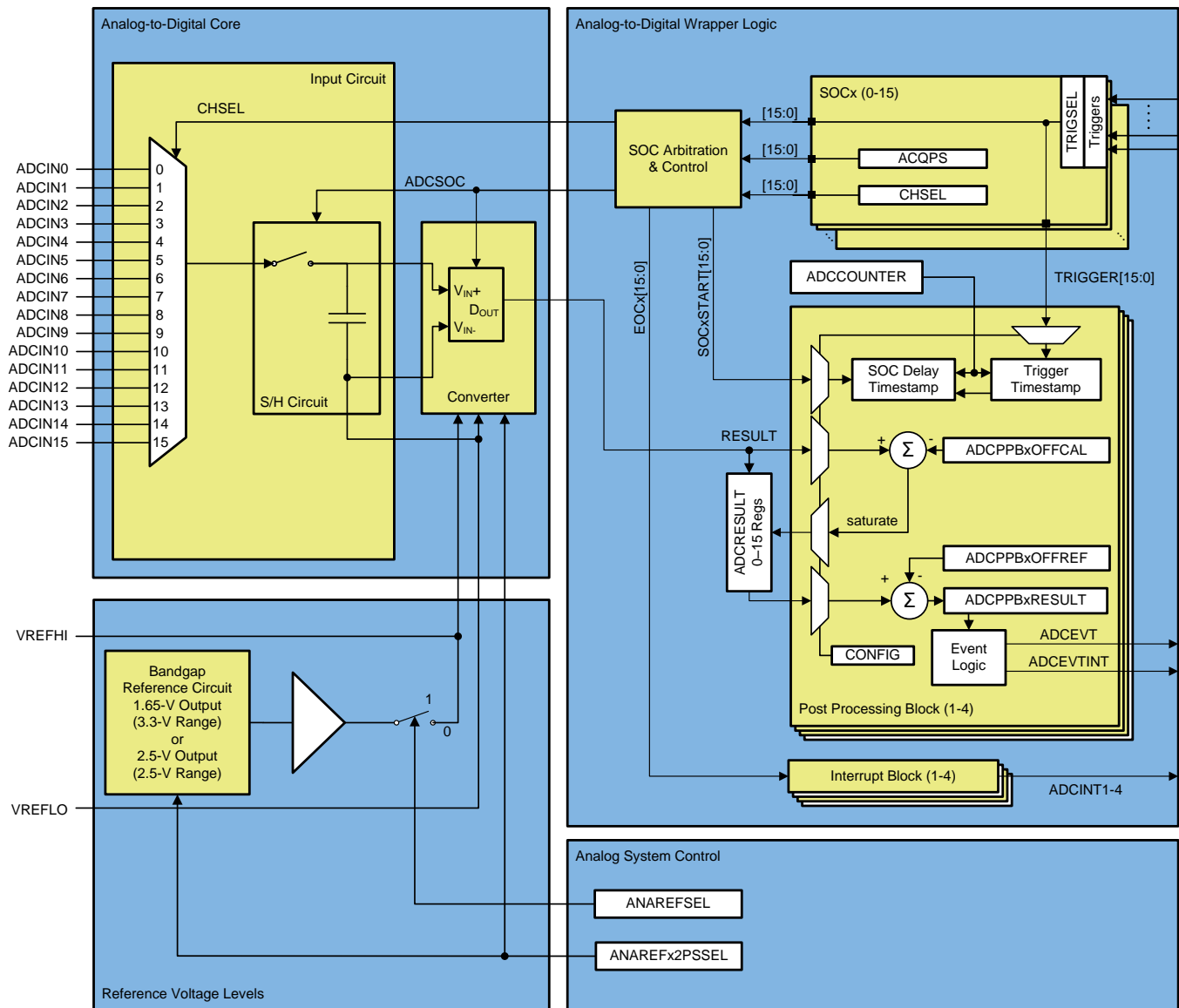
**NOTE:** Not every channel may be pinned out from all ADCs. Check the datasheet for your device to determine which channels are available.

---

### 13.3 ADC Block Diagram

The block diagram for the ADC core and ADC wrapper are presented in Figure 13-1.

Figure 13-1. ADC Module Block Diagram



### 13.4 ADC Configurability

Some ADC configurations are individually controlled by the SOCx, while others are globally controlled per ADC module. Table 13-1 summarizes the basic ADC options and their level of configurability. The subsequent sections discuss these configurations.

Table 13-1. ADC Options and Configuration Levels

Options	Configurability
Clock	Per module <sup>(1)</sup>
Resolution	Not configurable (12-bit only)

<sup>(1)</sup> Writing these values differently to different ADC modules could cause the ADCs to operate asynchronously. See Section 13.15.1 for guidance on when the ADCs are operating synchronously or asynchronously.

**Table 13-1. ADC Options and Configuration Levels (continued)**

Options	Configurability
Signal mode	Not configurable (single-ended only)
Reference voltage source	
Trigger source	Per SOC <sup>(1)</sup>
Converted channel	Per SOC
Acquisition window duration	Per SOC <sup>(1)</sup>
EOC location	Per module
Burst Mode	Per module <sup>(1)</sup>

### 13.4.1 Clock Configuration

The base ADC clock is provided directly by the system clock (SYSCLK). This clock is used to generate the ADC acquisition window. The register ADCCTL2 has a PRESCALE field which determines the ADCCLK. The ADCCLK is used to clock the converter.

The core requires approximately 10.5 ADCCLK cycles to process a voltage into a conversion result. It is necessary for the user to determine the required duration of the acquisition window, see [Section 13.15.2](#).

---

**NOTE:** To determine an appropriate value for ADCCTL2.PRESCALE, consult the datasheet of your device to determine the maximum SYSCLK and ADCCLK frequency.

---

### 13.4.2 Resolution

The resolution of the ADC determines how finely the analog range is quantized into digital values. This ADC supports a resolution of 12 bits.

### 13.4.3 Voltage Reference

#### 13.4.3.1 External Reference Mode

Each ADC has a VREFHI input and a VREFLO input. In external reference mode these pins are used as a ratiometric reference to determine the ADC conversion input range.

See [Section 13.15.4](#) for information on how to supply the reference voltage.

**NOTES:**

- On devices with no external VREFLO signals, VREFLO has been internally connected to the device analog ground, VSSA.
- Consult the datasheet for your device to determine the allowable voltage range for VREFHI and VREFLO.
- The external reference mode requires an external capacitor on the VREFHI pin. Consult the device datasheet for the specific value required.

#### 13.4.3.2 Internal Reference Mode

In internal reference mode the device drives a voltage out onto the VREFHI pin. The VREFHI and VREFLO pins then set the ADC conversion range.

The internal reference voltage can be configured to be either 2.5V or 1.65V. When a 1.65V internal reference voltage is selected, the effective ADC conversion input range will be VREFLO to 3.3V.

**NOTES:**

- The internal reference mode also requires an external capacitor on the VREFHI pin. Consult the device datasheet for the specific value required.

### 13.4.3.3 Ganged References

On some packages, the voltage reference pins for multiple ADCs may be combined. In this case, it is necessary to configure the ganged references identically when selecting external vs. internal reference mode and for selecting an internal reference voltage range of 3.3V or 2.5V.

For example, if ADC B and ADC C reference pins are combined and the desired reference mode is 2.5V internal reference mode, the following reference configuration code should be run:

```
//ADCB VREFHI and ADCC VREFHI share a pin
//ADCB VREFLO and ADCC VREFLO share a pin
//Both references must be explicitly configured
//Both references must be configured identically
SetVREF(ADC_ADCB, ADC_INTERNAL, ADC_VREF2P5);
SetVREF(ADC_ADCC, ADC_INTERNAL, ADC_VREF2P5);
```

Internal device hardware will ensure multiple references don't drive conflicting voltages onto the same pin. Because of this, references can be configured in any order or over any amount of time.

### 13.4.3.4 Selecting Reference Mode

The voltage reference mode should be configured by using either SetVREF or ADC\_setVREF function provided in C2000Ware. Using any of these functions ensures that the correct trim is loaded in the ADC trim registers. This function must be called at least once after a device reset. Do not configure the voltage reference mode by directly writing to the ANAREFCTL register.

### 13.4.4 Signal Mode

The ADC supports single-ended signaling.

In single-ended mode, the input voltage to the converter is sampled through a single pin (ADCINx), referenced to VREFLO.

### 13.4.5 Expected Conversion Results

Based on a given analog input voltage, the ideal expected digital conversion is given by the tables below. Fractional values are truncated.

**Table 13-2. Analog to 12-bit Digital Formulas**

Analog Input	Digital Result
when $ADCINy \leq VREFLO$	$ADCRESULTx = 0$
when $VREFLO < ADCINy < VREFHI$	$ADCRESULTx = 4096 \left( \frac{ADCINy - VREFLO}{VREFHI - VREFLO} \right)$
when $ADCINy \geq VREFHI$	$ADCRESULTx = 4095$

### 13.4.6 Interpreting Conversion Results

Based on a given ADC conversion result, the ideal corresponding analog input is given by the following tables. This corresponds to the center of the possible range of analog voltages that could have produced this conversion result.

**Table 13-3. 12-Bit Digital-to-Analog Formulas**

Digital Value	Analog Equivalent
when $ADCRESULTy = 0$	$ADCINx \leq VREFLO$
when $0 < ADCRESULTy < 4095$	$ADCINx = (VREFHI - VREFLO) \left( \frac{ADCRESULTy}{4096} \right) + VREFLO$
when $ADCRESULTy = 4095$	$ADCINx \geq VREFHI$

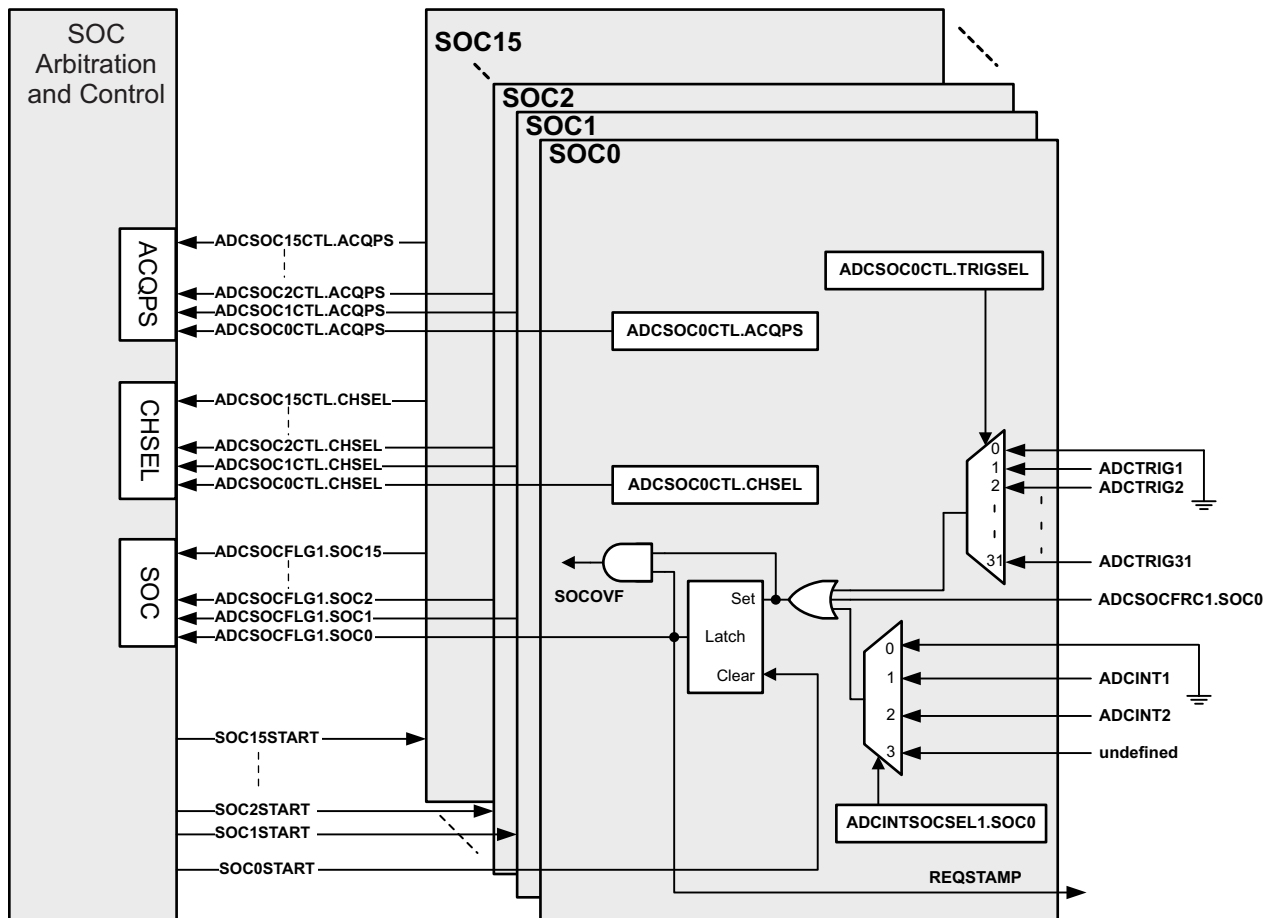


### 13.5 SOC Principle of Operation

The ADC triggering and conversion sequencing is accomplished through configurable start-of-conversions (SOCs). Each SOC is a configuration set defining the single conversion of a single channel. In that set there are three configurations: the trigger source that starts the conversion, the channel to convert, and the acquisition (sample) window duration. Upon receiving the trigger configured for a SOC, the wrapper will ensure that the specified channel is captured using the specified acquisition window duration.

Multiple SOCs can be configured for the same trigger, channel, and/or acquisition window as desired. Configuring multiple SOCs to use the same trigger will allow the trigger to generate a sequence of conversions. Configuring multiple SOCs to use the same trigger and channel will allow for oversampling.

Figure 13-2. SOC Block Diagram



#### 13.5.1 SOC Configuration

Each SOC has its own configuration register, ADCSOCxCTL. Within this register, SOCx can be configured for trigger source, channel to convert, and acquisition (sample) window duration.

#### 13.5.2 Trigger Operation

Each SOC can be configured to start on one of many input triggers. The primary trigger select for SOCx is in the ADCSOCxCTL.TRIGSEL register, which can select between:

- Disabled (software only)
- CPU Timers 0/1/2
- GPIO: Input X-Bar INPUT5
- ADCSOCA or ADCSOCB from each ePWM module

In addition, each SOC can also be triggered when the ADCINT1 flag or ADCINT2 flag is set. This is achieved by configuring the ADCINTSOCSEL1 register (for SOC0 to SOC7) or the ADCINTSOCSEL2 register (for SOC8 to SOC15). This is useful for creating continuous conversions.

### 13.5.3 ADC Acquisition (Sample and Hold) Window

External signal sources vary in their ability to drive an analog signal quickly and effectively. In order to achieve rated resolution, the signal source needs to charge the sampling capacitor in the ADC core to within 0.5LSBs of the signal voltage. The acquisition window is the amount of time the sampling capacitor is allowed to charge and is configurable for SOCx by the ADCSOCxCTL.ACQPS register.

ACQPS is a 9-bit register that can be set to a value between 0 and 511, resulting in an acquisition window duration of:

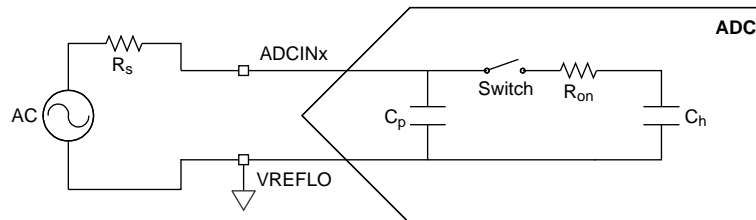
Acquisition window = (ACQPS + 1) · (System Clock (SYSCLK) cycle time)

- The acquisition window duration is based on the System Clock (SYSCLK), not the ADC clock (ADCCLK).
- The selected acquisition window duration must be at least as long as one ADCCLK cycle.
- The datasheet will specify a minimum acquisition window duration (in nanoseconds). The user is responsible for selecting an acquisition window duration that meets this requirement.

### 13.5.4 ADC Input Models

For single-ended operation, the ADC input characteristics for values in the single-ended input model (see [Figure 13-3](#)) can be found in the device data manual.

**Figure 13-3. Single-Ended Input Model**



These input models should be used along with actual signal source impedance to determine the acquisition window duration. See [Section 13.15.2](#) for more information.

### 13.5.5 Channel Selection

Each SOC can be configured to convert any of the ADC channels. This behavior is selected for SOCx by the ADCSOCxCTL.CHSEL register. This is summarized in [Table 13-4](#).

**Table 13-4. Channel Selection of Input Pins**

Input Mode	CHSEL	Input
Single-Ended	0	ADCIN0
	1	ADCIN1
	2	ADCIN2
	3	ADCIN3
	4	ADCIN4
	5	ADCIN5
	6	ADCIN6
	7	ADCIN7
	8	ADCIN8
	9	ADCIN9
	10	ADCIN10
	11	ADCIN11
	12	ADCIN12
	13	ADCIN13
	14	ADCIN14
	15	ADCIN15

## 13.6 SOC Configuration Examples

The following sections provide some specific examples of how to configure the SOCs to produce some conversions.

### 13.6.1 Single Conversion from ePWM Trigger

To configure ADCA to perform a single conversion on channel ADCINA1 when the ePWM timer reaches its period match, a few things are necessary. First, ePWM3 must be configured to generate an SOCA or SOCB signal (in this statement, SOC refers to a signal in the ePWM module). See the *Enhanced Pulse Width Modulator Module (ePWM)* chapter on how to do this. Assume that SOCB was chosen.

SOC5 is chosen arbitrarily. Any of the 16 SOCs could be used.

Assuming a 100ns sample window is desired with a SYSCLK frequency of 100MHz, then the acquisition window duration should be  $100\text{ns}/10\text{ns} = 10$  cycles. The ACQPS field should therefore be set to  $10 - 1 = 9$ .

```
AdcaRegs.ADCSOC5CTL.bit.CHSEL = 1;      //SOC5 will convert ADCINA1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 9;     //SOC5 will use sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC5CTL.bit.TRIGSEL = 10;  //SOC5 will begin conversion on ePWM3 SOCB
```

As configured, when ePWM3 matches its period and generates the SOCB signal, the ADC will begin sampling channel ADCINA1 (SOC5) immediately if the ADC is idle. If the ADC is busy, ADCINA1 will begin sampling when SOC5 gains priority (see [Section 13.7](#)). The ADC control logic will sample ADCINA1 with the specified acquisition window width of 100 ns. Immediately after the acquisition is complete, the ADC will begin converting the sampled voltage to a digital value. When the ADC conversion is complete, the results will be available in the ADCRESULT5 register (see [Section 13.14](#) for exact sample, conversion, and result latch timings).

### 13.6.2 Oversampled Conversion from ePWM Trigger

To configure the ADC to oversample ADCINA1 4 times, we use the same configurations as the previous example, but apply them to SOC5, SOC6, SOC7, and SOC8.

```
AdcaRegs.ADCSOC5CTL.bit.CHSEL = 1;      //SOC5 will convert ADCINA1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 9;     //SOC5 will use sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC5CTL.bit.TRIGSEL = 10;  //SOC5 will begin conversion on ePWM3 SOCB
AdcaRegs.ADCSOC6CTL.bit.CHSEL = 1;     //SOC6 will convert ADCINA1
AdcaRegs.ADCSOC6CTL.bit.ACQPS = 9;     //SOC6 will use sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC6CTL.bit.TRIGSEL = 10;  //SOC6 will begin conversion on ePWM3 SOCB
AdcaRegs.ADCSOC7CTL.bit.CHSEL = 1;     //SOC7 will convert ADCINA1
AdcaRegs.ADCSOC7CTL.bit.ACQPS = 9;     //SOC7 will use sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC7CTL.bit.TRIGSEL = 10;  //SOC7 will begin conversion on ePWM3 SOCB
AdcaRegs.ADCSOC8CTL.bit.CHSEL = 1;     //SOC8 will convert ADCINA1
AdcaRegs.ADCSOC8CTL.bit.ACQPS = 9;     //SOC8 will use sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC8CTL.bit.TRIGSEL = 10;  //SOC8 will begin conversion on ePWM3 SOCB
```

As configured, when ePWM3 matches its period and generates the SOCB signal, the ADC will begin sampling channel ADCINA1 (SOC5) immediately if the ADC is idle. If the ADC is busy, ADCINA1 will begin sampling when SOC5 gains priority (see [ADC Conversion Priority](#)). Once the conversion is complete for SOC5, SOC6 will begin converting ADCINA1 and the results for SOC5 will be placed in the ADCRESULT5 register. All four conversions will eventually be completed sequentially, with the results in ADCRESULT5, ADCRESULT6, ADCRESULT7, and ADCRESULT8 for SOC5, SOC6, SOC7, and SOC8, respectively.

---

**NOTE:** It is possible, but unlikely, that the ADC could begin converting SOC6, SOC7, or SOC8 before SOC5 depending on the position of the round-robin pointer when the ePWM trigger is received. See [ADC Conversion Priority](#) to understand how the next SOC to be converted is chosen.

---

### 13.6.3 Multiple Conversions from CPU Timer Trigger

This example will show how to sample multiple signals with different acquisition window requirements. CPU1 Timer 2 will be used to generate the trigger. To see how to configure the CPU timer, see the *System Control and Interrupts* chapter.

A good first step when designing a sampling scheme with many signals is to list out the signals and their required acquisition window. From this, calculate the necessary number of SYCLK cycles for each signal, then the ACQPS register setting. This is shown in [Table 13-5](#), where a SYCLK of 100MHz is assumed ( 10ns cycle time).

**Table 13-5. Example Requirements for Multiple Signal Sampling**

Signal Name	Acquisition Window Requirement (ns)	Acquisition Window SYCLK Cycles	ACQPS Register Value
Signal 1	>240ns	240ns/10ns = 24	24 – 1 = 23
Signal 2	>883.333ns	883.333ns/10ns = 89 (round up)	89 – 1 = 88
Signal 3	>220 ns	220ns/10ns = 22	22 – 1 = 21
Signal 4	>585ns	585ns/10ns = 59 (round up)	59 – 1 = 58

Next decide which ADC pins to connect to each signal. This will be highly dependent on application board layout. Once the pins are selected, determining the value of CHSEL is straightforward (see [Table 13-6](#)).

**Table 13-6. Example Connections for Multiple Signal Sampling**

Signal Name	ADC PIN	CHSEL Register Value
Signal 1	ADCINA5	5
Signal 2	ADCINA0	0
Signal 3	ADCINA3	3
Signal 4	ADCINA2	2

With the information tabulated, it is easy to generate the SOC configurations:

```
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 5;           //SOC0 will convert ADCINA5
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 23;         //SOC0 will use sample duration of 24 SYCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 3;        //SOC0 will begin conversion on CPU1 Timer 2
AdcaRegs.ADCSOC1CTL.bit.CHSEL = 0;         //SOC1 will convert ADCINA0
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 88;        //SOC1 will use sample duration of 89 SYCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 3;        //SOC1 will begin conversion on CPU1 Timer 2
AdcaRegs.ADCSOC2CTL.bit.CHSEL = 3;         //SOC2 will convert ADCINA3
AdcaRegs.ADCSOC2CTL.bit.ACQPS = 21;        //SOC2 will use sample duration of 22 SYCLK cycles
AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 3;        //SOC2 will begin conversion on CPU1 Timer 2
AdcaRegs.ADCSOC3CTL.bit.CHSEL = 2;         //SOC3 will convert ADCINA2
AdcaRegs.ADCSOC3CTL.bit.ACQPS = 58;        //SOC3 will use sample duration of 59 SYCLK cycles
AdcaRegs.ADCSOC3CTL.bit.TRIGSEL = 3;        //SOC3 will begin conversion on CPU1 Timer 2
```

As configured, when CPU1 Timer 2 generates an event, SOC0, SOC1, SOC2, and SOC3 will eventually be sampled and converted, in that order. The conversion results for ACINA5 (Signal 1) will be in ADCRESULT0. Similarly, The results for ADCINA0 (Signal 2), ADCINA3 (Signal 3), and ADCINA2 (Signal 4) will be in ADCRESULT1, ADCRESULT2, and ADCRESULT3, respectively.

---

**NOTE:** It is possible, but unlikely, that the ADC could begin converting SOC1, SOC2, or SOC3 before SOC0 depending on the position of the round-robin pointer when the CPU Timer trigger is received. See [ADC Conversion Priority](#) to understand how the next SOC to be converted is chosen.

---

### 13.6.4 Software Triggering of SOC's

At any point, whether or not the SOC's have been configured to accept a specific trigger, a software trigger can set the SOC's to be converted. This is accomplished by writing bits in the ADCSOCFRC1 register.

Software triggering of the previous example without waiting for the CPU1 Timer 2 to generate the trigger could be accomplished by the statement:

```
AdcaRegs.ADCSOCFRC1.all = 0x000F;           //set SOC flags for SOC0 to SOC3
```

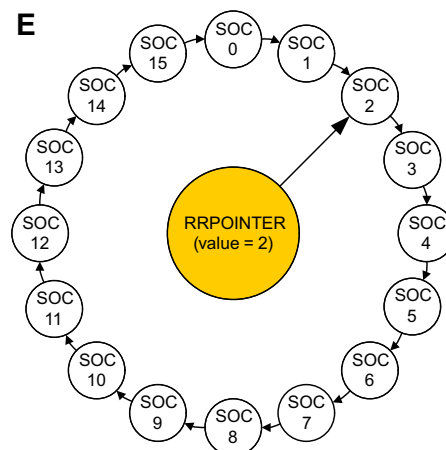
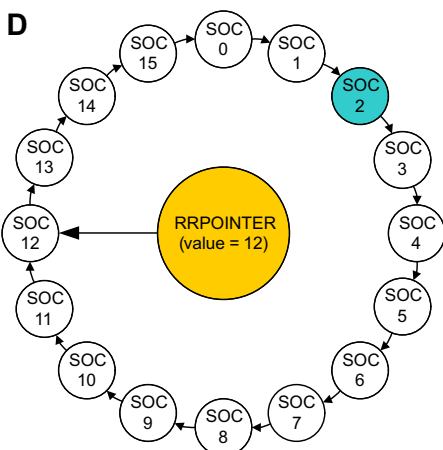
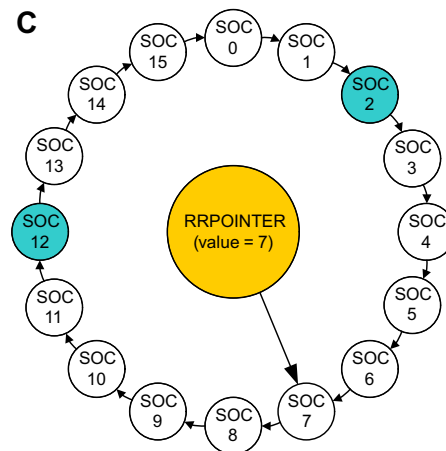
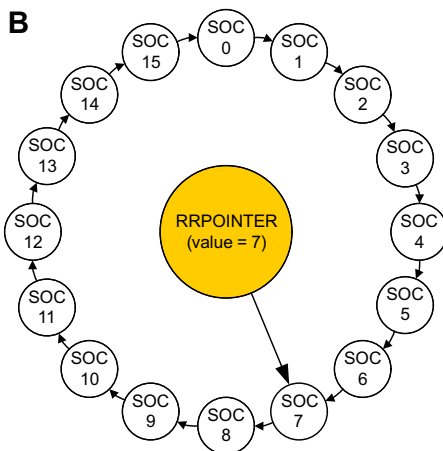
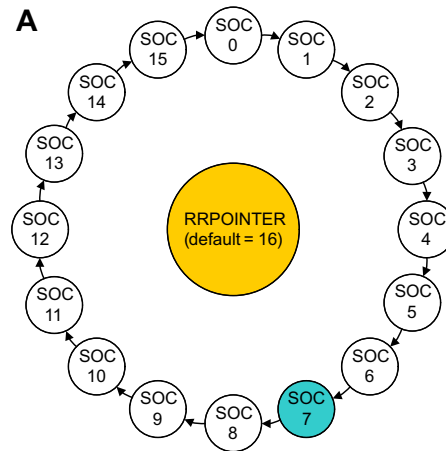
### 13.7 ADC Conversion Priority

When multiple SOC flags are set at the same time, one of two forms of priority determines the order in which they are converted. The default priority method is round robin. In this scheme, no SOC has an inherent higher priority than another. Priority depends on the round robin pointer (RRPOINTER). The RRPOINTER reflected in the ADCSOCPRIORITYCTL register points to the last SOC converted. The highest priority SOC is given to the next value greater than the RRPOINTER value, wrapping around back to SOC0 after SOC15. At reset the value is 16 since 0 indicates a conversion has already occurred. When RRPOINTER equals 16 the highest priority is given to SOC0. The RRPOINTER is reset by a device reset, when the ADCCTL1.RESET bit is set, or when the SOCPRICTL register is written.

An example of the round robin priority method is given in [Figure 13-4](#) .

Figure 13-4. Round Robin Priority Example

- A** After reset, SOC0 is highest priority SOC ; SOC7 receives trigger ; SOC7 configured channel is converted immediately .
- B** RRPOINTER changes to point to SOC 7 ; SOC8 is now highest priority SOC .
- C** SOC2 & SOC12 triggers rcvd . simultaneously ; SOC12 is first on round robin wheel ; SOC12 configured channel is converted while SOC2 stays pending .
- D** RRPOINTER changes to point to SOC 12 ; SOC2 configured channel is now converted .
- E** RRPOINTER changes to point to SOC 2 ; SOC3 is now highest priority SOC .



The SOC PRIORITY field in the ADCSOC PRIORITY CTL register can be used to assign high priority from a single to all of the SOC s. When configured as high priority, an SOC will interrupt the round robin wheel after any current conversion completes and insert itself in as the next conversion. After its conversion completes, the round robin wheel will continue where it was interrupted. If two high priority SOC s are triggered at the same time, the SOC with the lower number will take precedence.

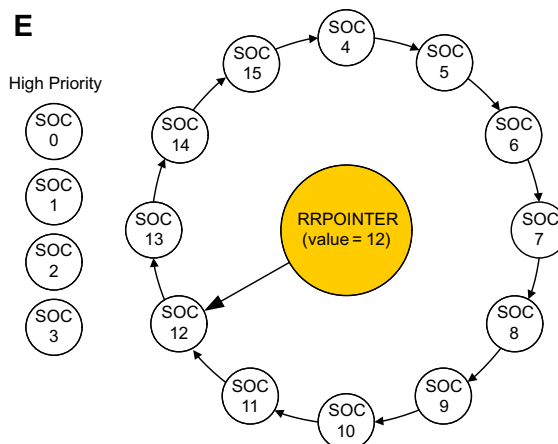
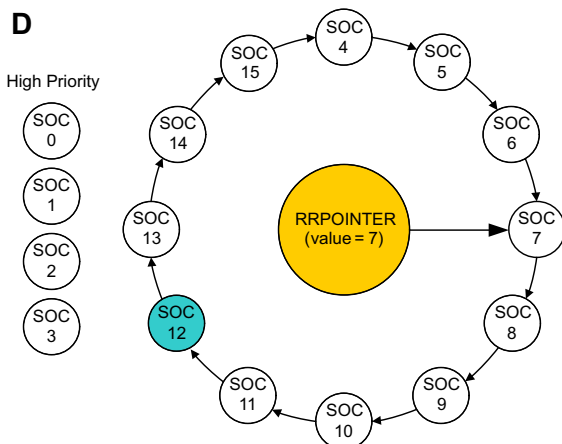
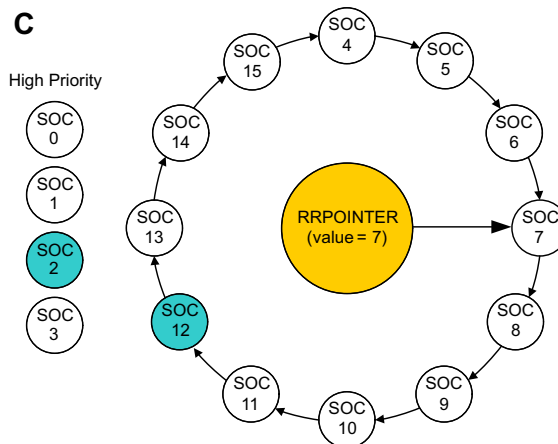
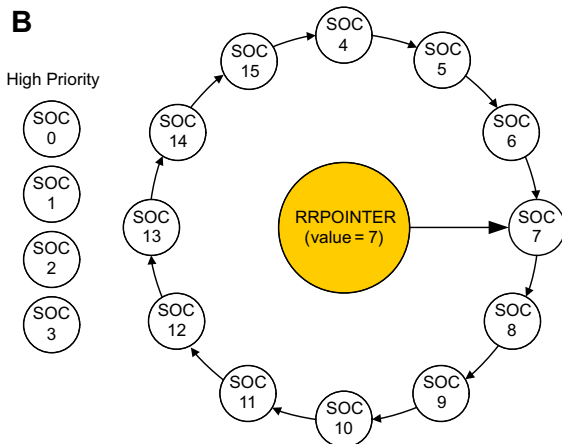
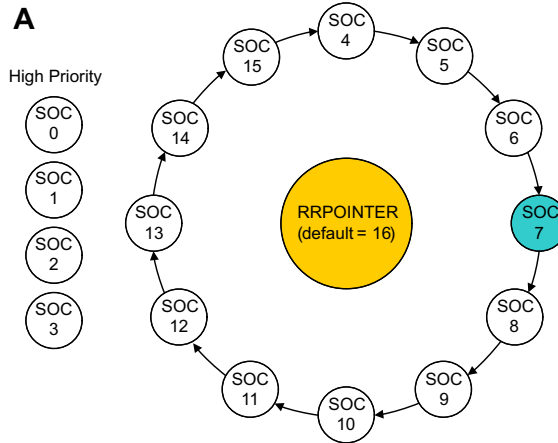
High priority mode is assigned first to SOC0, then in increasing numerical order. The value written in the SOC PRIORITY field defines the first SOC that is not high priority. In other words, if a value of 4 is written into SOC PRIORITY, then SOC0, SOC1, SOC2, and SOC3 are defined as high priority, with SOC0 the highest.

An example using high priority SOC's is given in [Figure 13-5](#).

**Figure 13-5. High Priority Example**

Example when SOC PRIORITY = 4

- A** After reset, SOC4 is 1<sup>st</sup> on round robin wheel ;  
SOC7 receives trigger ;  
SOC7 configured channel is converted immediately .
- B** RRPOINTER changes to point to SOC 7 ;  
SOC8 is now 1<sup>st</sup> on round robin wheel .
- C** SOC2 & SOC12 triggers rcvd . simultaneously ;  
SOC2 interrupts round robin wheel and SOC 2 configured channel is converted while SOC 12 stays pending .
- D** RRPOINTER stays pointing to 7 ;  
SOC12 configured channel is now converted .
- E** RRPOINTER changes to point to SOC 12 ;  
SOC13 is now 1<sup>st</sup> on round robin wheel .





## 13.8 Burst Mode

Burst mode allows a single trigger to walk through the round-robin SOC's one or more at a time. Setting the bit BURSTEN in the ADCBURSTCTL register configures the ADC wrapper for burst mode. This causes the TRIGSEL field to be ignored, but only for SOC's that are configured for round-robin operation (not high priority). Instead of the TRIGSEL field, all round-robin SOC's are triggered based on the BURSTTRIG field in the ADCBURSTCTL register. Upon reception of the burst trigger, the ADC wrapper will not set all round-robin SOC's to be converted, but only (ADCBURSTCTL.BURSTSIZE + 1) SOC's. The first SOC to be set will be that with the highest priority based on the round-robin pointer, and subsequent SOC's will be set until BURSTSIZE SOC's have been set.

---

**NOTE:** When configuring the ADC for burst mode, the user is responsible for ensuring that each burst of conversions is allowed to complete before the next burst trigger is received.

---

### 13.8.1 Burst Mode Example

Burst mode can be used to sample a different set of signals on every other trigger. In the following example, ADCIN7 and ADCIN5 are converted on the first trigger from CPU1 Timer 2 and every other trigger thereafter. ADCIN2 and ACIN3 are converted on the second trigger from CPU1 Timer 2 and every other trigger thereafter. All signals are converted with 20 SYSCLK cycle wide acquisition windows, but different durations could be configured for each SOC as desired.

```
AdcaRegs.BURSTCTL.BURSTEN = 1;           //Enable ADC burst mode
AdcaRegs.BURSTCTL.BURSTTRIG = 3;        //CPU1 Timer 2 will trigger burst of conversions
AdcaRegs.BURSTCTL.BURSTSIZE = 1;       //conversion bursts are 1 + 1 = 2 conversions long

AdcaRegs.SOCPRICL.bit.SOCPRIORITY = 12; //SOC0 to SOC11 are high priority

AdcaRegs.ADCSOC12CTL.bit.CHSEL = 7;     //SOC12 will convert ADCINA7
AdcaRegs.ADCSOC12CTL.bit.ACQPS = 19;    //SOC12 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC13CTL.bit.CHSEL = 5;     //SOC13 will convert ADCINA5
AdcaRegs.ADCSOC13CTL.bit.ACQPS = 19;    //SOC13 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC14CTL.bit.CHSEL = 2;     //SOC14 will convert ADCINA2
AdcaRegs.ADCSOC14CTL.bit.ACQPS = 19;    //SOC14 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC15CTL.bit.CHSEL = 3;     //SOC15 will convert ADCINA3
AdcaRegs.ADCSOC15CTL.bit.ACQPS = 19;    //SOC15 will use sample duration of 20 SYSCLK cycles
```

When the first CPU1 Timer 2 trigger is received, SOC12 and SOC13 will be converted immediately if the ADC is idle. If the ADC is busy, SOC12 and SOC13 will be converted once their SOC's gain priority. The results for SOC12 and SOC13 will be in ADCRESULT12 and ADCRESULT13, respectively. After SOC13 completes, the round robin pointer will give highest priority to SOC14. Because of this, when the next CPU1 Timer 2 trigger is received, SOC14 and SOC15 will be set as pending and eventually be converted. The results for SOC14 and SOC15 will be in ADCRESULT14 and ADCRESULT15, respectively. Subsequent triggers will continue to toggle between converting SOC12 and SOC13, and converting SOC14 and SOC15.

While the above example toggles between two sets of conversions, three or more different sets of conversions could be achieved using a similar approach.

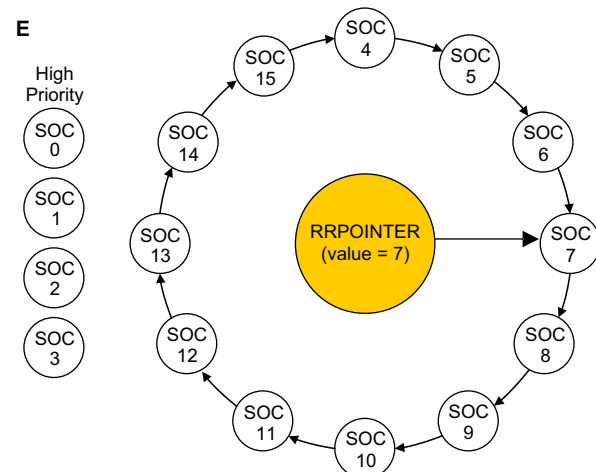
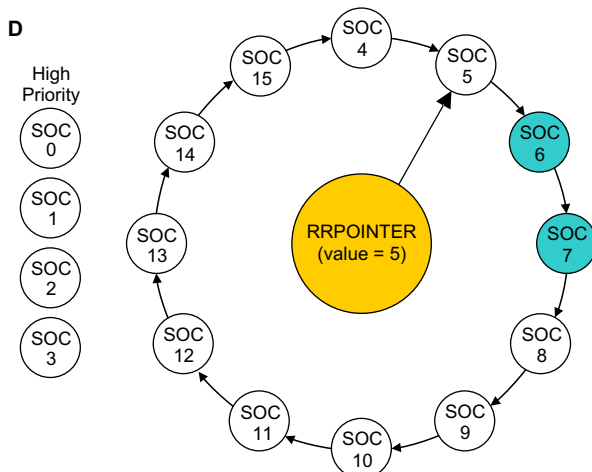
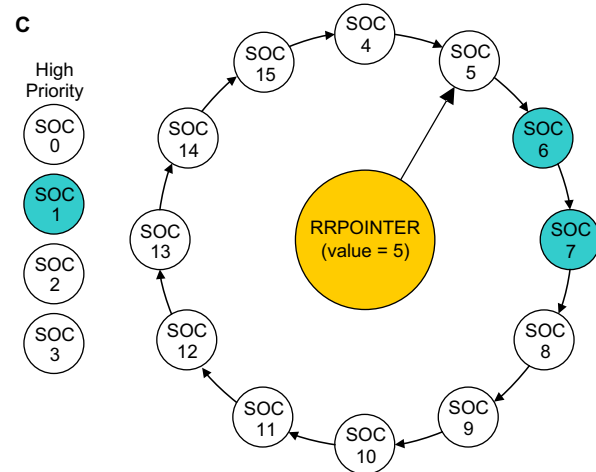
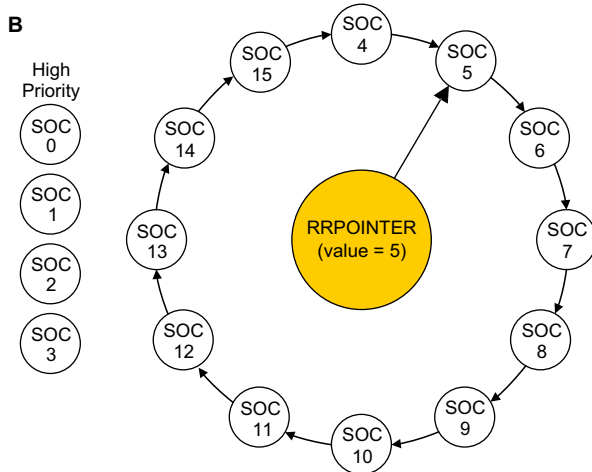
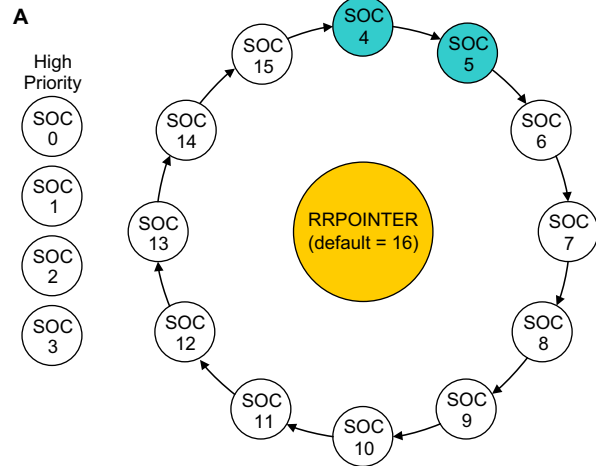
### 13.8.2 Burst Mode Priority Example

An example of priority resolution using burst mode and high-priority SOC's is presented in [Figure 13-6](#).

Figure 13-6. Burst Priority Example

Example when SOC PRIORITY = 4, BURSTEN = 1, and BURSTSIZE = 1

- A** After reset, SOC4 is 1<sup>st</sup> on round robin wheel; BURSTTRIG trigger is received; SOC4 & SOC5 are set and configured channels converted immediately.
- B** RRPOINTER changes to point to SOC5; SOC6 is now 1<sup>st</sup> on round robin wheel.
- C** BURSTTRIG & SOC1 triggers rcvd. simultaneously; SOC1, SOC6, and SOC7 are set; SOC1 interrupts round robin wheel and SOC1 configured channel is converted while SOC6 and SOC7 stay pending.
- D** RRPOINTER stays pointing to 5; SOC6/SOC7 configured channels are now converted.
- E** RRPOINTER changes to point to SOC7; SOC8 is now 1<sup>st</sup> on round robin wheel, waiting for BURSTTRIG.



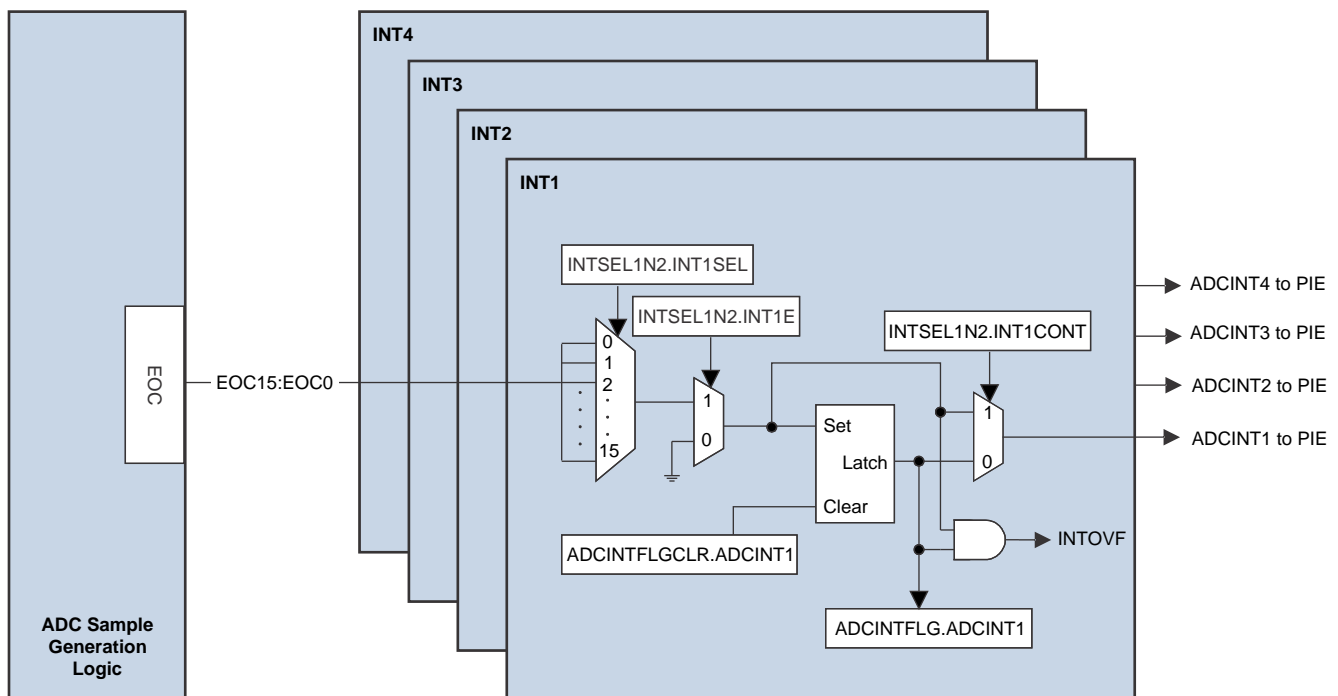
### 13.9 EOC and Interrupt Operation

Each SOC has a corresponding end-of-conversion (EOC) signal. This EOC signal can be used to trigger an ADC interrupt. The ADC can be configured to generate the EOC pulse at either the end of the acquisition window or at the end of the voltage conversion. This is configured using the bit INTPULSEPOS in the ADCCTL1 register. See Section 13.14, for exact EOC pulse location.

Each ADC module has 4 configurable ADC interrupts. These interrupts can be triggered by any of the 16 EOC signals. The flag bit for each ADCINT can be read directly or the interrupt can be passed on to the PIE.

Figure 13-7 shows a block diagram of the ADC interrupt structure.

Figure 13-7. ADC EOC Interrupts



#### 13.9.1 Interrupt Overflow

If the EOC signal would set a flag in the ADCINTFLG register, but that flag is already set, an interrupt overflow occurs. By default, overflow interrupts will not be passed on to the PIE module. When an overflow occurs on a given flag in the ADCINTFLG register, the corresponding flag in the ADCINOVF register is set. This overflow flag is only used to detect that an overflow has occurred; it does not block further interrupts from propagating to the PIE module.

When an ADC interrupt overflow could occur, the application should check the appropriate ADCINTOVF flag inside the ISR or in the background loop and take appropriate action when an overflow is detected. The following code snippet demonstrates how to check the ADCINOVF flag inside the ISR after attempting to clear the ADCINT flag.

```

AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1; //clear INT1 flag for ADC-A
if(1 == AdcaRegs.ADCINTOVF.bit.ADCINT1) //ADCINT overflow occurred
{
    AdcaRegs.ADCINTOVFCLR.bit.ADCINT1 = 1 //Clear overflow flag
    AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1 //Re-clear ADCINT flag
}
    
```

### 13.9.2 Continue to Interrupt Mode

The INTxCONT bits in the ADCINTSEL1N2 and ADCINTSEL3N4 registers configure how interrupts are handled when an ADCINTFLG has not yet been cleared from a prior interrupt. This mode is disabled by default and additional overlapping interrupts will not be issued to the PIE. By activating this mode ADC interrupts will always reach the PIE. The ADCINTOVF register will still be set if an interrupt occurs while ADCINTFLG is still set regardless of this configuration.

### 13.9.3 Early Interrupt Configuration Mode

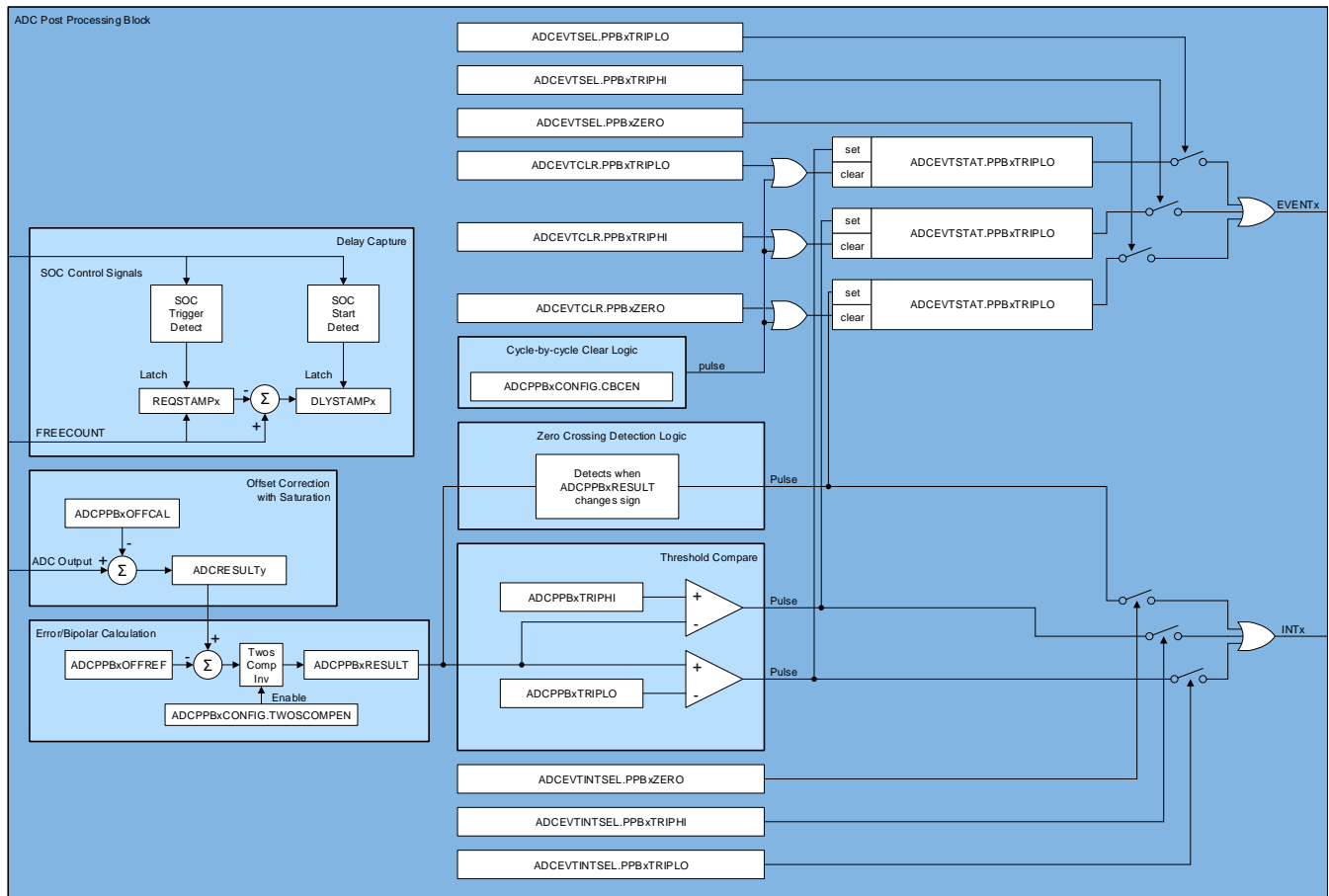
Enabling early interrupt mode can allow the application to enter the ADC interrupt service routine before the ADC results are ready. This allows the application to do any necessary pre-work so that the application can act on the ADC results immediately when they become available. However, the timing of the early interrupt may be too early and the application may need to waste time until the updated ADC results become available. To prevent this situation, the time the ADC interrupt is entered in early interrupt mode is configurable via the OFFSET field in the ADCINTCYCLE register.

- To use the configurable interrupt time, the ADC must be in early interrupt mode. To achieve this, clear the bit INTPULSEPOS to 0 in ADCCTL1.
- The OFFSET value of the ADCINTCYCLE register sets the number of additional SYSCLK cycles after the falling edge of the SOC pulse before the ADCINT flag is set.
- If the value of OFFSET goes beyond EOC, the ADC interrupt will be generated along with EOC.
- Writing values to OFFSET when INTPULSEPOS is set to 1 will not have any effect on the interrupt generation.

### 13.10 Post-Processing Blocks

Each ADC module contains four post-processing blocks (PPB). These blocks can be associated with any of the 16 RESULT registers using the ADCPPBxCONFIG.CONFIG bit field. Each PPB can simultaneously remove an offset associated with the ADCIN channel, subtract out a reference value, flag a zero-crossing point, and flag a high or low compare limit. Furthermore, the zero-crossing and compare flags can trip a PWM and/or generate an interrupt. A PPB is also capable of recording the delay between when the SOC associated with the PPB is triggered and when it actually begins to be sampled. [Figure 13-8](#) presents the structure of each PPB. Subsequent sections explain the use of each submodule.

Figure 13-8. ADC PPB Block Diagram



### 13.10.1 PPB Offset Correction

In many applications, external sensors and signal sources produce an offset. A global trimming of the ADC offset is not enough to compensate for these offsets, which vary from channel to channel. The post-processing block can remove these offsets with zero overhead, saving numerous cycles in tight control loops.

Offset correction is accomplished by first pointing the ADCPPBxCONFIG.CONFIG to the desired SOC, then writing an offset correction value to the ADCPPBxOFFCAL.OFFCAL register. The post-processing block will automatically add or subtract the value in the OFFCAL register from the raw conversion result and store it in the ADCRESULT register. This addition/subtraction will saturate at 0 on the low end and 4095 on the high end.

**NOTES:**

- Writing a 0 to the OFFCAL register effectively disables the offset correction feature, passing the raw result unchanged to the ADCRESULT register.
- It is possible to point multiple PPBs to the same SOC. In this case, the OFFCAL value that will actually be applied will be that of the PPB with the highest number.
- In particular, care needs to be taken when using the PPB on SOC0, as all the PPB point to this SOC by default. This may cause unintentional overwriting of offset correction of a lower numbered PPB by a higher numbered PPB.

### 13.10.2 PPB Error Calculation

In many applications, an error from a setpoint or expected value must be computed from the digital output of an ADC conversion. In other cases, a bipolar signal is necessary or convenient for control calculations. The PPB can perform these function automatically, reducing the sample to output latency and reducing software overhead.

Error calculation is accomplished by first pointing the ADCPPBxCONFIG.CONFIG to the desired SOC, then writing a value to the ADCPPBxOFFCAL.OFFREF register. The post-processing block will automatically subtract the value in the OFFREF register from the ADCRESULT value and store it in the ADCPPBxRESULT register. This subtraction will produce a sign-extended 32-bit result. It is also possible to selectively invert the calculated value before storing in the ADCPPBxRESULT register by setting the TWOSCOMPEN bit in the ADCPPBxCONFIG register.

#### NOTES:

- Do not write a value larger than 12 bits to the OFFREF register.
- Since the PPBxRESULT register is unique for each PPB, it is possible to point multiple PPBs to the same SOC and get different results for each PPB.
- Writing a 0 to the OFFREF register effectively disables the error calculation feature, passing the ADCRESULT value unchanged to the ADCPPBxRESULT register.

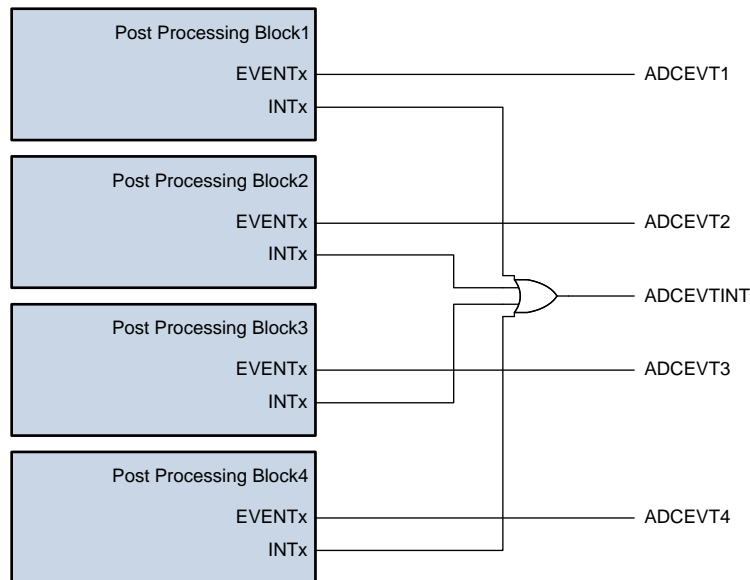
### 13.10.3 PPB Limit Detection and Zero-Crossing Detection

Many applications perform a limit check against the ADC conversion results. The PPB can automatically perform a check against a high and low limit or whenever ADCPPBxRESULT changes sign. Based on these comparisons, it can generate a trip to the PWM and/or an interrupt automatically, lowering the sample to ePWM latency and reducing software overhead. This functionality also enables safety conscious applications to trip the ePWM based on an out-of-range ADC conversion without any CPU intervention.

To enable this functionality, first point the ADCPPBxCONFIG.CONFIG to the desired SOC, then write a value to one or both of the registers ADCPPBxTRIPHI.LIMITHI and ADCPPBxTRIPLO.LIMITLO (zerocrossing detection does not require further configuration). Whenever these limits are exceeded, the PPBxTRIPHI bit or PPBxTRIPLO bit will be set in the ADCEVTSTAT register. Note that the PPBxZERO bit in the ADCEVTSTAT register is gated by EOC and not by the sign change in the ADCPPBxRESULT register. The ADCEVTCLR register has corresponding bits to clear these event flags. The ADCEVTSEL register has corresponding bits which allow the events to propagate through to the PWM. The ADCINTSEL register has corresponding bits which allow the events to propagate through to the PIE.

One PIE interrupt is shared between all the PPBs for a given ADC module as shown in [Figure 13-9](#).

**Figure 13-9. ADC PPB Interrupt Event**



**NOTES:**

- Zero-crossing and limit compare reference the ADCPPBxRESULT register. This will include any correction applied by the OFFCAL and OFFREF registers. TRIPHI and TRIPLO do NOT perform a signed comparison. It is recommended to leave OFFREF as 0 when using limit compare functionality.
- If different actions need to be taken for different PPB events from the same ADC module, then the ADCEVTINT ISR will have to read the PPB event flags in the ADCEVTSTAT register to determine which event caused the interrupt.
- If different ePWM trips need to be generated separately for high compare, low compare, and/or zero-crossing, this can be achieved by pointing multiple PPBs to the same SOC.

**13.10.4 PPB Sample Delay Capture**

When multiple control loops are running asynchronously on the same ADC, there is a chance that an ADC request from two or more loops will collide, causing one of the samples to be delayed. This shows up as a measurement error in the system. By knowing when this delay occurs and the amount of delay that has occurred software can employ extrapolation techniques to reduce the error.

To this effect, each PPB has the field DLYSTAMP in the ADCPPBxSTAMP register. This field will contain the number of SYSCLK cycles between when the associate SOC was triggered and when it began converting.

This is achieved by having a global 12-bit free running counter based off of SYSCLK, which is in the field FREECOUNT in the ADCCOUNTER register. When the trigger for the associated SOC arrives, the value of this counter is loaded into the bit field ADCPPBxTRIPLO.REQSTAMP. When the actual sample window for that SOC begins, the value in REQSTAMP is subtracted from the current FREECOUNT value and stored in DLYSTAMP.

---

**NOTE:** If more than 4096 SYSCLK cycles elapse between the SOC trigger and the actual start of the SOC acquisition, the FREECOUNT register may overflow more than once, leading to incorrect DLYSTAMP value. Be cautious when using very slow conversions to prevent this from happening.

---



---

**NOTE:** The sample delay capture will not function if the associated SOC is triggered via software. It will, however, correctly record the delay if the software triggering of a different SOC causes the SOC associated with the PPB to be delayed

---



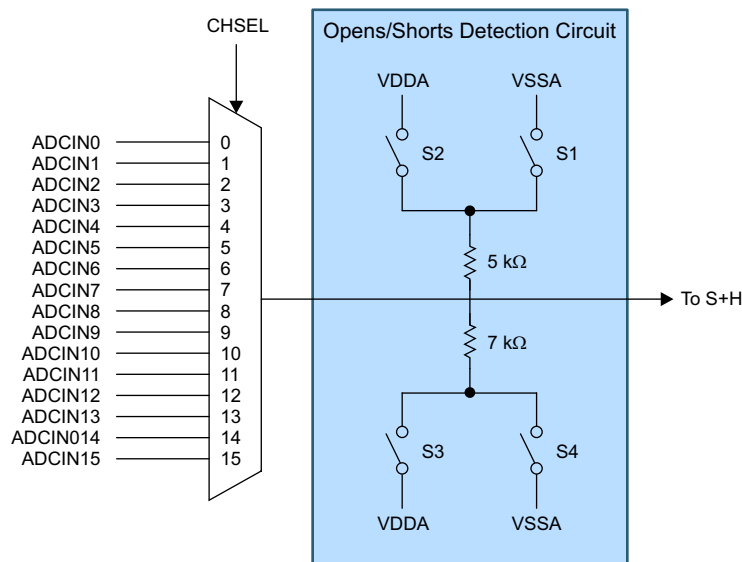
### 13.11 Opens/Shorts Detection Circuit (OSDETECT)

The opens/shorts detection circuit (OSDETECT) can be used to detect pin faults in the system. The circuit connects to the ADC input after the channel select multiplexer but before the S+H circuit as shown in [Figure 13-10](#).

**NOTES:**

- The divider resistance tolerances can vary widely, hence this feature should not be used to check for conversion accuracy.
- Consult the device data manual for implementation and availability of analog input channels.
- Due to high drive impedance, a S+H duration much longer than the ADC minimum will be needed .

**Figure 13-10. Opens/Shorts Detection Circuit**



The circuit can be operated by writing a value to the DETECTCFG field in the ADCOSDETECT register. This will cause the circuit to source a voltage onto the input during the S+H phase of any conversion. The voltage and drive strength of the OSDETECT circuit for different DETECTCFG settings is given in [Table 13-7](#).

**Table 13-7. DETECTCFG Settings**

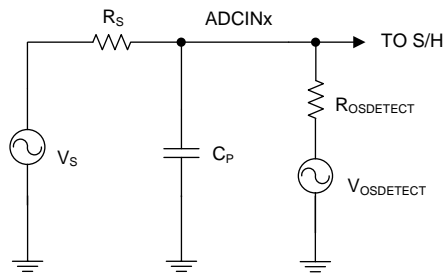
ADCOSDETECT.DETECTCFG	Source Voltage	S4	S3	S2	S1	Drive Impedance
0	Off	Open	Open	Open	Open	Open
1	Zero Scale	Closed	Open	Open	Closed	5K    7K
2	Full Scale	Open	Closed	Closed	Open	5K    7K
3	5/12 VDDA	Open	Closed	Open	Closed	5K    7K
4	7/12 VDDA	Closed	Open	Closed	Open	5K    7K
5	Zero Scale	Open	Open	Open	Closed	5K
6	Full Scale	Open	Open	Closed	Open	5K
7	Zero Scale	Closed	Open	Open	Open	7K



### 13.11.1 Implementation

A representative circuit with the OSDETECT implementation consists of the signal source with series resistance  $R_S$ , shunt capacitor  $C_P$ , the equivalent OSDETECT resistance  $R_{OSDETECT}$  and voltage  $V_{OSDETECT}$  is shown in Figure 13-11 and can be used as a basis to calculate the signal level going in to the sampling capacitor.  $R_{OSDETECT}$  and  $V_{OSDETECT}$  are the equivalent input resistance and voltage source contributed by the OSDETECT circuit with values shown in Table 13-7 for the different configuration settings. Refer to Figure 13-11 when deriving the input signal to S/H if signal source  $V_S$  is driving while the OSDETECT feature is enabled.

Figure 13-11. Input Circuit Equivalent with OSDETECT Enabled



The input impedance  $R_S$  and  $C_P$  may be integral parts of the signal source or these could have been implemented in the design to precondition the signal or to control signal settling time to meet S/H requirements. The input path has to be considered when using the OSDETECT feature as this would affect the conversion results. For instance, driving an input signal when this feature is enabled would connect signal  $V_S$  to the OSDETECT circuit through  $R_S$  and affecting the ADC results. Larger  $C_P$  values (in the order greater than hundreds of pF) would require using higher ACQPS to ensure that signal at the input has settled prior to conversion.

To enable the circuit:

1. Configure the ADC for conversion (for example, channel, SOC, ACQPS, prescaler, trigger, and so on).
2. Set up the ADCOSDETECT register for the desired voltage divider connection as shown in Table 13-7.
3. Initiate a conversion and inspect the conversion result.

**Note:** You must interpret the results based on what is driving on the input side and what are the values of  $R_S$  and  $C_P$ . If the  $V_S$  signal can be disconnected from the input pin, the circuit can be used to detect open and shorted input pins as described in the following sections.

first then

### 13.11.2 Detecting an Open Input Pin

By cycling through the various OSDETECT settings, the input signal will be pulled towards the sourced voltages. An input with good drive strength (pin not open) will be minimally affected. However, if the pin is open, the sampled voltages will be close to the source voltages specified in Table 13-7.

### 13.11.3 Detecting a Shorted Input Pin

By cycling through the various OSDETECT settings, the input signal will be pulled towards the sourced voltages. An input with finite drive strength (pin not shorted) will be pulled toward each sourced voltage. However, if the pin is shorted, the signal will remain at the same voltage.

## 13.12 Power-Up Sequence

Upon device power-up or system level reset, the ADC will be powered down and disabled. When powering up the ADC, use the following sequence:

1. Set the bit to enable the desired ADC clock in the PCLKCR13 register.
2. Set the desired ADC clock divider in the PRESCALE field of ADCCTL2.
3. Power up the ADC by setting the ADCPWDNZ bit in ADCCTL1.
4. Allow a delay before sampling. See the data manual for the necessary time.

If multiple ADCs are powered up simultaneously, steps 1 and step 3 can each be done for all ADCs in one write instruction. Also, only one delay is necessary as long as it occurs after all the ADCs have begun powering up.

### 13.13 ADC Calibration

During the fabrication and test process, Texas Instruments calibrates the gain, offset, and linearity of the ADCs and the offset of the buffered DACs and the gain and offset of the PGAs. These trim settings are embedded into TI reserved OTP memory as part of C-callable functions.

- The Device\_cal() function copies the trim values for ADC and PGA gain and offset from OTP memory to their respective trim registers.
- The trim functions in Device\_cal() are callable in C2000Ware as ADC\_setOFFSETTRIM() and ADC\_setINLTRIM() and PGA\_setGAINTRIM(). These functions fetch trim values from the TI reserved OTP memory source locations where the values are stored during test process as well as the analog module register destinations where the trim values are copied to.

Until the appropriate factory trim is loaded, the ADC (and other modules) will not be guaranteed to operate within datasheet specifications. Similarly, if trim values other than the factory settings are placed into the trim registers, the ADC (and other modules) will not be guaranteed to operate within datasheet specifications.

The boot ROM will call the calibration functions, so trim values should be initially populated without user intervention. However, if the trims are cleared due to a module reset or modified for some other reason, then the user can call the calibration functions (defined in the headerfiles).

#### 13.13.1 ADC Zero Offset Calibration

Zero offset error is defined as the difference from 0 that occurs when converting a voltage at VREFLO. The zero offset error can be positive or negative. To correct this error, an adjustment of equal magnitude and opposite polarity is written into the ADCOFFTRIM register. The value contained in this register will be applied before the results are available in the ADC result registers. This operation is fully contained within the ADC core, so the timing of the results will not be affected and the full dynamic range of the ADC will be maintained for any trim value.

Using the GetAdcOffsetTrimOTP(Uint16) function, the ADCOFFTRIM register can be loaded with the factory calibrated offset error correction. The user can modify the ADCOFFTRIM register to compensate for additional offset error induced by the application environment if desired, but this is not typically necessary to achieve datasheet specified performance.

---

**NOTE:** Regardless of the converter resolution, the size of each ADCOFFTRIM step is  $(VREFHI - VREFLO)/65536$ .

---

Use the following procedure to re-calibrate the ADC offset in 12-bit, single-ended mode:

1. Set ADCOFFTRIM to +112 steps (0x70). This adds an artificial offset to account for negative offset that may reside in the ADC core.
2. Perform some multiple of 16 conversions on VREFLO (internal connection), accumulating the results (for example,  $32 \times 16$  conversions = 512 conversions).
3. Divide the accumulated result by the multiple of 16 (for example, for 512 conversions, divide by 32).
4. Set ADCOFFTRIM to  $112 - \text{result from step 3}$ .

### 13.14 ADC Timings

The process of converting an analog voltage to a digital value is broken down into an S+H phase and a conversion phase. The ADC sample and hold circuits (S+H) are clocked by SYSCLK while the ADC conversion process is clocked by ADCCLK. ADCCLK is generated by dividing down SYSCLK based on the PRESCALE field in the ADCCTL2 register.

The S+H duration is the value of the ACQPS field of the SOC being converted, plus one, times the SYSCLK period. The user must ensure that this duration exceeds both 1 ADCCLK period and the minimum S+H duration specified in the datasheet. The conversion time is approximately 10.5 ADCCLK cycles. The exact conversion time is always a whole number of SYSCLK cycles. See the timing diagrams and tables in [Section 13.14.1](#) for exact timings.

### 13.14.1 ADC Timing Diagrams

The following diagrams show the ADC conversion timings for two SOC's given the following assumptions:

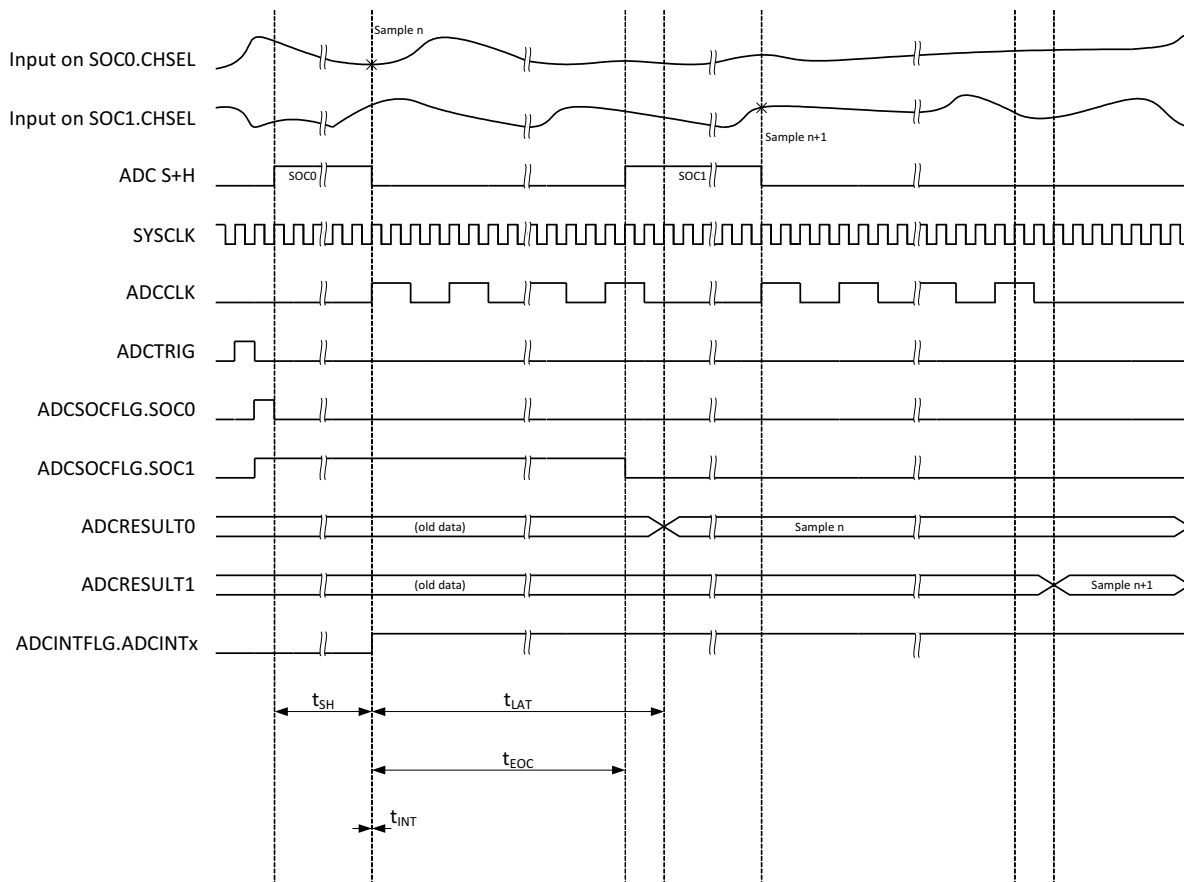
- SOC0 and SOC1 are configured to use the same trigger.
- No other SOC's are converting or pending when the trigger occurs.
- The round robin pointer is in a state that causes SOC0 to convert first.
- ADCINTSEL is configured to set an ADCINT flag upon end of conversion for SOC0 (whether this flag propagates through to the CPU to cause an interrupt is determined by the configurations in the PIE module).

The following parameters are identified in the timing diagrams:

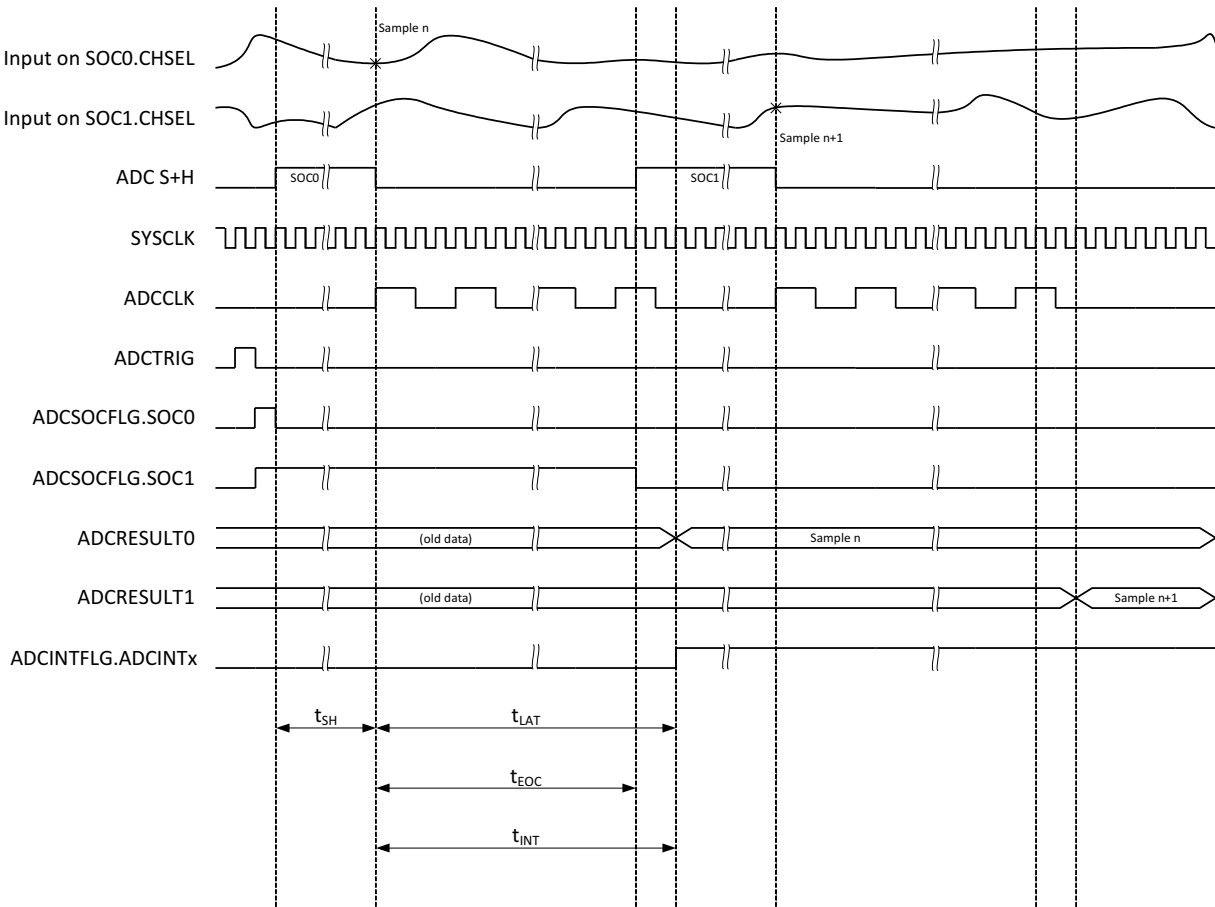
**Table 13-8. ADC Timing Parameters**

PARAMETER	DESCRIPTION
$t_{SH}$	<p>The duration of the S+H window.</p> <p>At the end of this window, the value on the S+H capacitor becomes the voltage to be converted into a digital value. The duration is given by <math>(ACQPS + 1)</math> SYSCLK cycles. ACQPS can be configured individually for each SOC, so <math>t_{SH}</math> will not necessarily be the same for different SOCs.</p> <p><b>Note:</b> The value on the S+H capacitor will be captured approximately 5ns before the end of the S+H window regardless of device clock settings.</p>
$t_{LAT}$	<p>The time from the end of the S+H window until the ADC results latch in the ADCRESULTx register.</p> <p>If the ADCRESULTx register is read before this time, the previous conversion results will be returned.</p>
$t_{EOC}$	<p>The time from the end of the S+H window until the S+H window for the next ADC conversion can begin. The subsequent sample can start before the conversion results are latched. The subsequent sample can start before the conversion results are latched.</p>
$t_{INT}$	<p>The time from the end of the S+H window until an ADCINT flag is set (if configured).</p> <p>If the INTPULSEPOS bit in the ADCCTL1 register is set, <math>t_{INT}</math> will coincide with the conversion results being latched into the result register.</p> <p>If the INTPULSEPOS bit is 0, <math>t_{INT}</math> will coincide with the end of the S+H window. If <math>t_{INT}</math> triggers a read of the ADC result register (directly through DMA or indirectly by triggering an ISR that reads the result), care must be taken to ensure the read occurs after the results latch (otherwise, the previous results will be read).</p> <p>If the INTPULSEPOS bit is 0, and the OFFSET field in the ADCINTCYCLE register is not 0, then there will be a delay of OFFSET SYSCLK cycles before the ADCINT flag is set. This delay can be used to enter the ISR or trigger the DMA at exactly the time the sample is ready.</p>

**Figure 13-12. ADC Timings for 12-bit Mode in Early Interrupt Mode**



**Figure 13-13. ADC Timings for 12-bit Mode in Late Interrupt Mode**



**Table 13-9. ADC Timings in 12-bit Mode (SYSCLK Cycles)**

ADCCTL2.PRESCALE	Prescale Ratio	$t_{EOC}$	$t_{LAT}$	$t_{INT}$ (Early)	$t_{INT}$ (Late)
0	1	11	12	0	12
2	2	21	22	0	22
4	3	31	33	0	33
6	4	41	43	0	43
8	5	51	54	0	54
10	6	61	64	0	64
12	7	71	75	0	75
14	8	81	85	0	85

### 13.15 Additional Information

The following sections contain additional practical information.

#### 13.15.1 Ensuring Synchronous Operation

For best performance, all ADCs on the device should be operated synchronously. The device datasheet specifies the performance in both synchronous and asynchronous mode for those parameters which differ between the modes of operation.

To ensure synchronous operation, all ADCs on the device should operate in lockstep. This is accomplished by writing configurations to all ADCs that cause the sampling and conversion phases of all ADCs to be exactly aligned. The easiest way to accomplish this is to write identical values to the SOC configurations for each ADC for trigger select and ACQPS (S+H duration).

### 13.15.1.1 Basic Synchronous Operation

The below example configures two SOC's each on ADCA and ADCB with identical trigger select and ACQPS values. This will result in synchronous operation between ADCA and ADCB. For devices with more than two ADCs, the same principles can be used to synchronize all the ADCs.

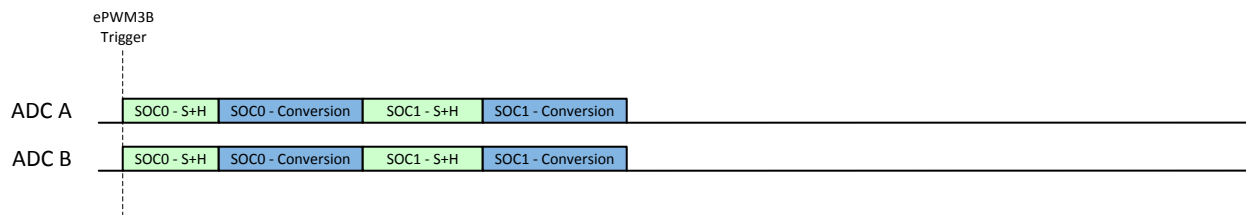
**Example: Basic Synchronous Operation**

```

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4; //SOC0 will convert ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0; //SOC0 will convert ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4; //SOC1 will convert ADCINA4
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1; //SOC1 will convert ADCINB1
AdcbRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 will use sample duration of 31 SYSCLK cycles
AdcbRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB
    
```

**Figure 13-14. Example: Basic Synchronous Operation**



Several things should be noted from [Figure 13-14](#). First, while the ACQPS values must be the same for SOC's with the same number, different ACQPS values can be used for SOC's with different numbers. Because of this, synchronous operation does not require a single global S+H time, but instead only channels sampled simultaneously require identical S+H durations. Another important point from this example is that any channel select value can be used for any SOC. Finally, this example assumes round-robin operation. If high priority SOC's are to be used, the priority must be configured the same on all ADCs.

### 13.15.1.2 Synchronous Operation with Multiple Trigger Sources

As long as each set of SOC's has identical trigger select and ACQPS settings, multiple trigger sources can be used while still achieving synchronous operation.

The below example demonstrates synchronous operation between ADCA and ADCB while using three SOC's and two trigger sources. [Figure 13-15](#) demonstrates that any combination of relative trigger timings still results in synchronous operation.

**Example: Synchronous Operation With Multiple Trigger Sources**

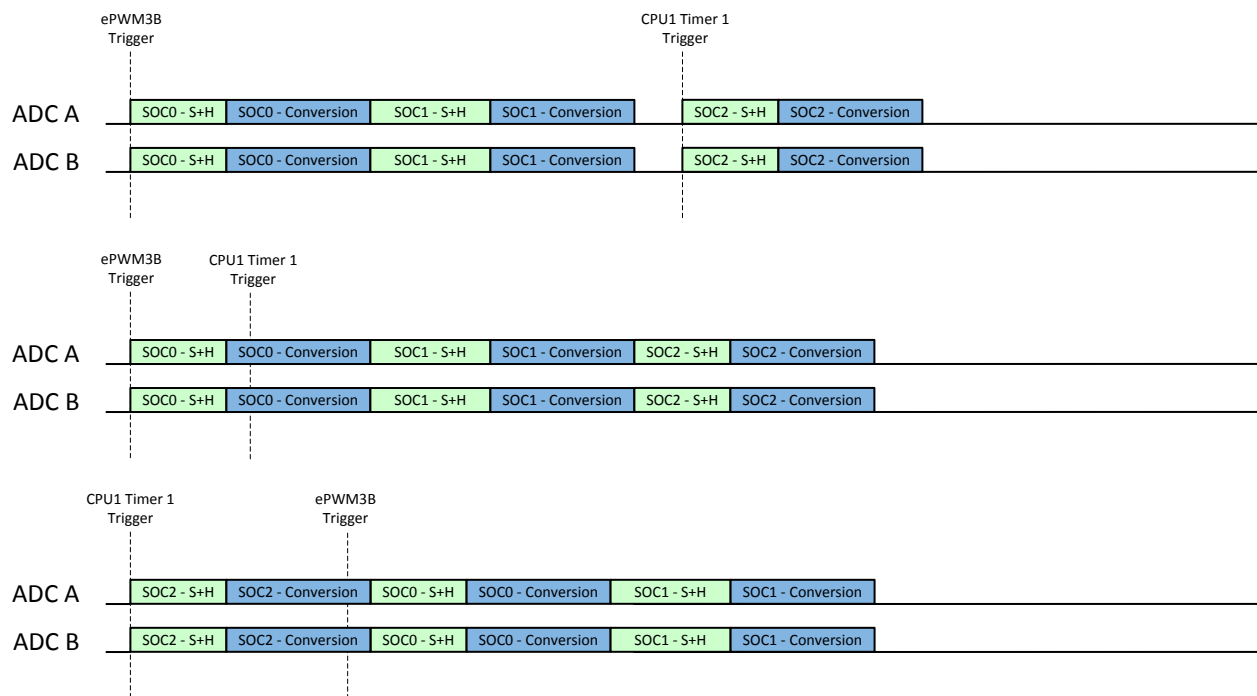
```

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4; //SOC0 will convert ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0; //SOC0 will convert ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4; //SOC1 will convert ADCINA4
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1; //SOC1 will convert ADCINB1
AdcbRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 will use sample duration of 31 SYSCLK cycles
AdcbRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB

AdcaRegs.ADCSOC2CTL.bit.CHSEL = 0; //SOC2 will convert ADCINA0
AdcaRegs.ADCSOC2CTL.bit.ACQPS = 19; //SOC2 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 2; //SOC2 will begin conversion on CPU Timer1
AdcbRegs.ADCSOC2CTL.bit.CHSEL = 2; //SOC2 will convert ADCINB2
AdcbRegs.ADCSOC2CTL.bit.ACQPS = 19; //SOC2 will use sample duration of 31 SYSCLK cycles
AdcbRegs.ADCSOC2CTL.bit.TRIGSEL = 2; //SOC2 will begin conversion on CPU Timer1
    
```

**Figure 13-15. Example: Synchronous Operation with Multiple Trigger Sources**



Note that any trigger source that can be selected in the TRIGSEL field can be used except for software triggering. There is no way to issue the software triggers for all ADCs simultaneously, so it will likely result in asynchronous operation. ADCINT1 or ADCINT2 can also be used as a trigger as long as the ADCINTSOCSEL1 and ADCINTSOCSEL2 registers are configured identically for all ADCs and software triggering is not used to start the chain of conversions.

**13.15.1.3 Synchronous Operation with Uneven SOC Numbers**

If only one trigger source is used, one ADC can use more SOC's than the other ADCs while still operating synchronously.



**Example: Synchronous Operation With Uneven SOC Numbers**

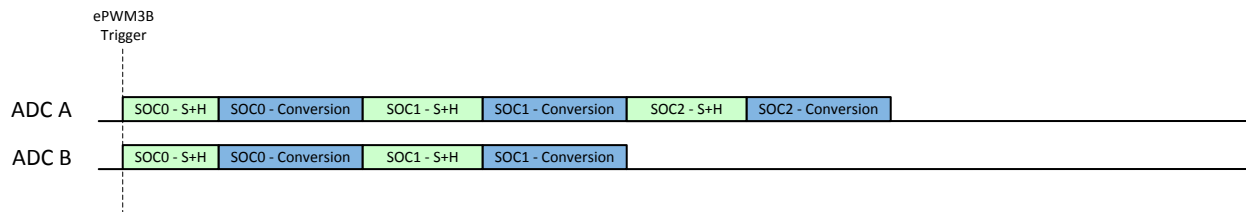
```

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4; //SOC0 will convert ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0; //SOC0 will convert ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4; //SOC1 will convert ADCINA4
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1; //SOC1 will convert ADCINB1
AdcbRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 will use sample duration of 31 SYSCLK cycles
AdcbRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB

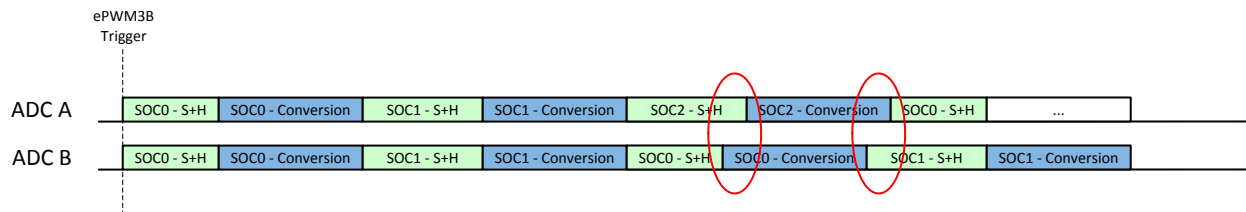
AdcaRegs.ADCSOC2CTL.bit.CHSEL = 0; //SOC2 will convert ADCINA0
AdcaRegs.ADCSOC2CTL.bit.ACQPS = 19; //SOC2 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 10; //SOC2 will begin conversion on ePWM3 SOCB
    
```

**Figure 13-16. Example: Synchronous Operation with Uneven SOC Numbers**



Note that if the trigger comes again before all SOCs have completed their conversions, ADCB will begin converting immediately on SOC0 while ADCA will not start converting SOC0 again until SOC2 is complete. This will result in asynchronous operation, so care must be taken to not overflow the trigger.

**Figure 13-17. Example: Asynchronous Operation with Uneven SOC Numbers – Trigger Overflow**



**13.15.1.4 Non-overlapping Conversions**

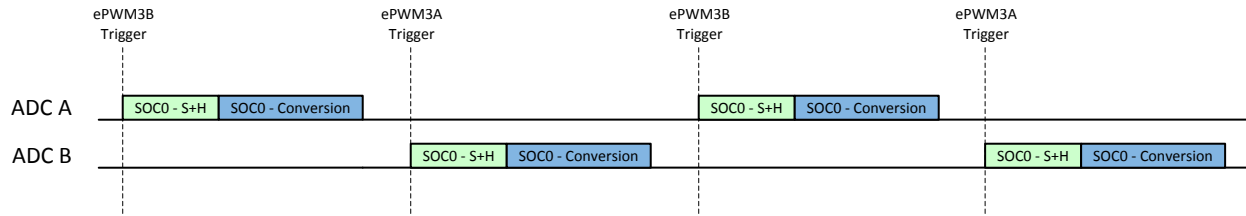
If conversion timings can be guaranteed to not overlap by the user, then it is not necessary to configure all SOCs identically on all ADCs to achieve performance equivalent to synchronous operation. For example, if the two ADC triggers in a system come from two ePWM sources which are always 180 degrees out-of-phase, then SOC0 could be used for both ADCA and ADCB with different trigger sources and different ACQPS values.

**Example: Operation with Non-overlapping Conversions**

```

//ePWM3 SOCA and SOCB are 180 degrees out of phase
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4; //SOC0 will convert ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0; //SOC0 will convert ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 9; //SOC0 will begin conversion on ePWM3 SOCA
    
```



**Figure 13-18. Example: Synchronous Equivalent Operation with Non-Overlapping Conversions**


### 13.15.2 Choosing an Acquisition Window Duration

For correct operation, the input signal to the ADC must be allowed adequate time to charge the sample and hold capacitor,  $C_h$ . Typically, the S+H duration is chosen such that the sampling capacitor will be charged to within  $\frac{1}{2}$  LSB or  $\frac{1}{4}$  LSB of the final value, depending on the tolerable settling error.

An approximation of the required settling time can be determined using an RC settling model. The time constant for the model is given by the equation:

$$\tau = (R_s + R_{on}) \cdot C_h + R_s \cdot (C_s + C_p)$$

And the number of time constants needed is given by the equation:

$$k = \ln\left(\frac{2^n}{\text{settling error}}\right) - \ln\left(\frac{C_s + C_p}{C_h}\right)$$

So the total S+H time should be set to at least:

$$t = k \cdot \tau$$

Where the following parameters are provided by the ADC input model in the device data manual:

- $n$  = ADC resolution (in bits)
- $R_{ON}$  = ADC sampling switch resistance (in Ohms)
- $C_H$  = ADC sampling capacitor (in pF)
- $C_p$  = ADC channel parasitic input capacitance (in pF)

And the following parameters are dependent on the application design:

- settling error = tolerable settling error (in LSBs)
- $R_s$  = ADC driving circuit source impedance (in Ohms)
- $C_s$  = capacitance on ADC input pin (in pF)

For example, assuming the following parameters:

- $n$  = 12-bits
- $R_{ON}$  = 500 $\Omega$
- $C_H$  = 12.5pF
- $C_p$  = 12.7pF
- settling error =  $\frac{1}{4}$  LSB
- $R_s$  = 180 $\Omega$
- $C_s$  = 150pF

The time constant would be calculated as:

$$\tau = (180\Omega + 500\Omega) \cdot 12.5\text{pF} + 180\Omega \cdot (150\text{pF} + 12.7\text{pF}) = 8.5\text{ns} + 29.3\text{ns} = 37.8\text{ns}$$

And the number of required time constants would be:

$$k = \ln\left(\frac{2^{12}}{0.25}\right) - \ln\left(\frac{150\text{pF} + 12.7\text{pF}}{12.5\text{pF}}\right) = 9.70 - 2.57 = 7.13$$

So the S+H time should be set to at least:

$$37.8\text{ns} \cdot 7.13 = 270\text{ns}$$

If SYSCLK = 100 MHz, then each SYSCLK is 10ns. S+H duration should be 270ns/10ns = 27.0 SYSCLKs so ACQPS for this input should be set to at least CEILING(27.0) – 1 = 26.

While this gives a rough estimate of the required acquisition window, a better method would be to setup a circuit with the ADC input model, a model of the source impedance/capacitance, and any board parasitics in SPICE (or similar software) and simulate to verify that the sampling capacitor settles to the desired accuracy.

---

**NOTE:** The device datasheet will specify a minimum ADC S+H window duration. Do not use an ACQPS value that gives a duration less than this specification.

---

### 13.15.3 Achieving Simultaneous Sampling

While each ADC does not have dual S+H circuits, it is easy to achieve simultaneous sampling. This is accomplished by setting the SOC triggers on two or more ADC modules to use the same trigger source. The example below demonstrates x3 simultaneous sampling based on an ePWM3 event. ADCINA3, ADCINB5, and ADCINC5 are sampled. An acquisition window of 20 SYSCLK cycles is used, but different durations are possible.

```
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 3;           //SOC0 will convert ADCINA3
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19;          //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10;        //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 5;           //SOC0 will convert ADCINB5
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;          //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10;        //SOC0 will begin conversion on ePWM3 SOCB
AdccRegs.ADCSOC0CTL.bit.CHSEL = 5;           //SOC0 will convert ADCINC5
AdccRegs.ADCSOC0CTL.bit.ACQPS = 19;          //SOC0 will use sample duration of 20 SYSCLK cycles
AdccRegs.ADCSOC0CTL.bit.TRIGSEL = 10;        //SOC0 will begin conversion on ePWM3 SOCB
```

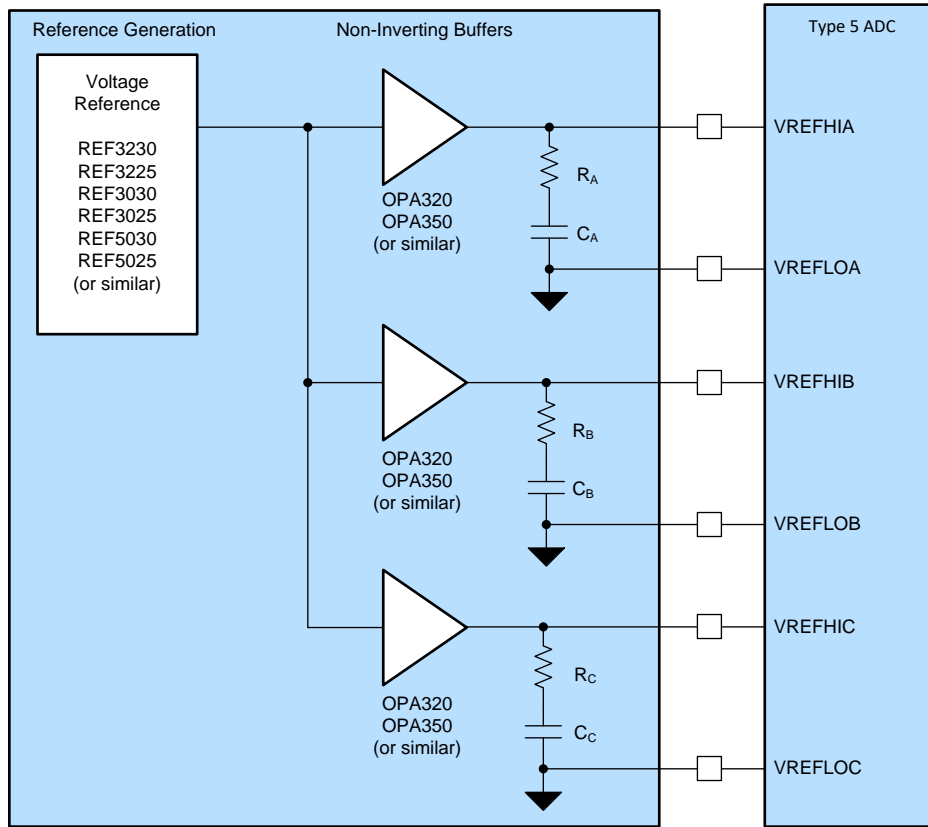
When the ePWM3 trigger is received, all three ADCs will begin converting in parallel immediately. All results will be in the ADCRESULT0 register for each ADC. Note that this assumes that all ADCs are idle when the trigger is received. If one or more ADCs is busy, the samples will not happen at exactly the same time.

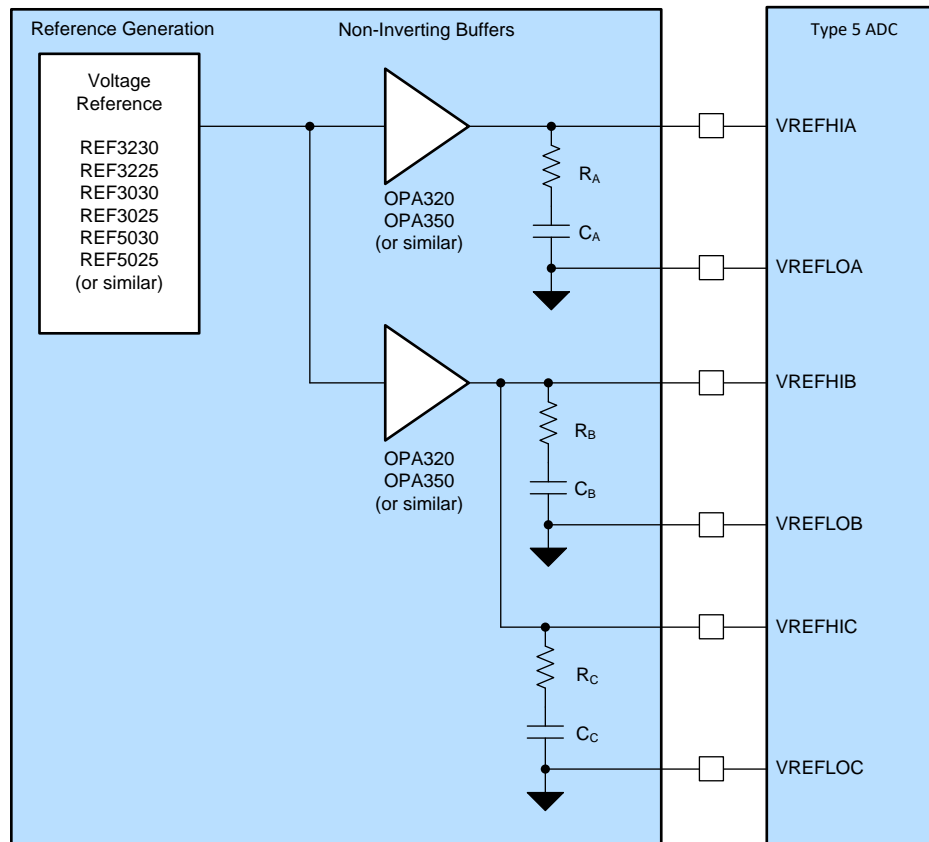
### 13.15.4 Designing an External Reference Circuit

shows the basic organization of the external voltage reference generation circuitry. A single reference voltage generation source should be shared by all ADC modules. This will minimize reference voltage mismatch between ADC modules. The reference voltage should then be buffered by a precision op-amp with good bandwidth and low output impedance before being driven into the reference pin. A capacitor between the high and low reference pins should be placed on the PCB as close to the pins as practical to help absorb high frequency currents. A series resistor (typically <math><1\Omega</math>) in series with this capacitor may be necessary to ensure op-amp stability.

It is also possible to share two reference pins between one op-amp driver. This organization is shown in . This will give slightly reduced performance compared to the case where each reference pin has a dedicated op-amp buffer, but it should still be possible to achieve all ADC specifications in the data manual.

Figure 13-19. ADC Shared Reference System



**Figure 13-20. ADC Shared Reference**


### 13.15.5 Internal Temperature Sensor

The internal temperature sensor measures the junction temperature of the device. The output of the sensor can be sampled with the ADC through an internal connection. This can be enabled on channel ADCIN14 on ADCB by setting the ENABLE bit in the TSNSCTL register.

To convert the temperature sensor reading into a temperature, pass the temperature sensor reading to the `GetTemperatureC()` function in `F28004x_TempSensorConv.c`.

Note that this function assumes that the temperature reading was taken with  $VREFHI = 2.5V$ . If a different reference voltage is used, the sample should be scaled appropriately before passing it to the function by using the below formula.

$$\text{adjusted sensor reading} = \text{raw sensor reading} * (VREFHI / 2.5V)$$

## 13.16 ADC Registers

This section describes the Analog-to-Digital Converter Registers.

### 13.16.1 ADC Base Addresses

**Table 13-10. ADC Base Address Table**

Device Registers	Register Name	Start Address	End Address
AdcaResultRegs	ADC_RESULT_REGS	0x0000_0B00	0x0000_0B1F
AdcbResultRegs	ADC_RESULT_REGS	0x0000_0B20	0x0000_0B3F
AdccResultRegs	ADC_RESULT_REGS	0x0000_0B40	0x0000_0B5F
AdcaRegs	ADC_REGS	0x0000_7400	0x0000_747F
AdcbRegs	ADC_REGS	0x0000_7480	0x0000_74FF
AdccRegs	ADC_REGS	0x0000_7500	0x0000_757F

### 13.16.2 ADC\_REGS Registers

Table 13-11 lists the ADC\_REGS registers. All register offset addresses not listed in Table 13-11 should be considered as reserved locations and the register contents should not be modified.

**Table 13-11. ADC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	ADCCTL1	ADC Control 1 Register	EALLOW	<a href="#">Go</a>
1h	ADCCTL2	ADC Control 2 Register	EALLOW	<a href="#">Go</a>
2h	ADCBURSTCTL	ADC Burst Control Register	EALLOW	<a href="#">Go</a>
3h	ADCINTFLG	ADC Interrupt Flag Register		<a href="#">Go</a>
4h	ADCINTFLGCLR	ADC Interrupt Flag Clear Register		<a href="#">Go</a>
5h	ADCINTOVF	ADC Interrupt Overflow Register		<a href="#">Go</a>
6h	ADCINTOVFCLR	ADC Interrupt Overflow Clear Register		<a href="#">Go</a>
7h	ADCINTSEL1N2	ADC Interrupt 1 and 2 Selection Register	EALLOW	<a href="#">Go</a>
8h	ADCINTSEL3N4	ADC Interrupt 3 and 4 Selection Register	EALLOW	<a href="#">Go</a>
9h	ADCSOCPRCTL	ADC SOC Priority Control Register	EALLOW	<a href="#">Go</a>
Ah	ADCINTSOCSEL1	ADC Interrupt SOC Selection 1 Register	EALLOW	<a href="#">Go</a>
Bh	ADCINTSOCSEL2	ADC Interrupt SOC Selection 2 Register	EALLOW	<a href="#">Go</a>
Ch	ADCSOCFLG1	ADC SOC Flag 1 Register		<a href="#">Go</a>
Dh	ADCSOCFRC1	ADC SOC Force 1 Register		<a href="#">Go</a>
Eh	ADCSOCOVF1	ADC SOC Overflow 1 Register		<a href="#">Go</a>
Fh	ADCSOCOVFCLR1	ADC SOC Overflow Clear 1 Register		<a href="#">Go</a>
10h	ADCSOC0CTL	ADC SOC0 Control Register	EALLOW	<a href="#">Go</a>
12h	ADCSOC1CTL	ADC SOC1 Control Register	EALLOW	<a href="#">Go</a>
14h	ADCSOC2CTL	ADC SOC2 Control Register	EALLOW	<a href="#">Go</a>
16h	ADCSOC3CTL	ADC SOC3 Control Register	EALLOW	<a href="#">Go</a>
18h	ADCSOC4CTL	ADC SOC4 Control Register	EALLOW	<a href="#">Go</a>
1Ah	ADCSOC5CTL	ADC SOC5 Control Register	EALLOW	<a href="#">Go</a>
1Ch	ADCSOC6CTL	ADC SOC6 Control Register	EALLOW	<a href="#">Go</a>
1Eh	ADCSOC7CTL	ADC SOC7 Control Register	EALLOW	<a href="#">Go</a>
20h	ADCSOC8CTL	ADC SOC8 Control Register	EALLOW	<a href="#">Go</a>
22h	ADCSOC9CTL	ADC SOC9 Control Register	EALLOW	<a href="#">Go</a>
24h	ADCSOC10CTL	ADC SOC10 Control Register	EALLOW	<a href="#">Go</a>
26h	ADCSOC11CTL	ADC SOC11 Control Register	EALLOW	<a href="#">Go</a>
28h	ADCSOC12CTL	ADC SOC12 Control Register	EALLOW	<a href="#">Go</a>
2Ah	ADCSOC13CTL	ADC SOC13 Control Register	EALLOW	<a href="#">Go</a>
2Ch	ADCSOC14CTL	ADC SOC14 Control Register	EALLOW	<a href="#">Go</a>
2Eh	ADCSOC15CTL	ADC SOC15 Control Register	EALLOW	<a href="#">Go</a>
30h	ADCEVTSTAT	ADC Event Status Register		<a href="#">Go</a>
32h	ADCEVTCLR	ADC Event Clear Register		<a href="#">Go</a>
34h	ADCEVTSEL	ADC Event Selection Register	EALLOW	<a href="#">Go</a>
36h	ADCEVTINTSEL	ADC Event Interrupt Selection Register	EALLOW	<a href="#">Go</a>
38h	ADCOSDETECT	ADC Open and Shorts Detect Register	EALLOW	<a href="#">Go</a>
39h	ADCCOUNTER	ADC Counter Register		<a href="#">Go</a>
3Ah	ADCREV	ADC Revision Register		<a href="#">Go</a>
3Bh	ADCOFFTRIM	ADC Offset Trim Register	EALLOW	<a href="#">Go</a>
40h	ADCPPB1CONFIG	ADC PPB1 Config Register	EALLOW	<a href="#">Go</a>
41h	ADCPPB1STAMP	ADC PPB1 Sample Delay Time Stamp Register		<a href="#">Go</a>
42h	ADCPPB1OFFCAL	ADC PPB1 Offset Calibration Register	EALLOW	<a href="#">Go</a>
43h	ADCPPB1OFFREF	ADC PPB1 Offset Reference Register		<a href="#">Go</a>

**Table 13-11. ADC\_REGS Registers (continued)**

Offset	Acronym	Register Name	Write Protection	Section
44h	ADCPPB1TRIPHI	ADC PPB1 Trip High Register	EALLOW	<a href="#">Go</a>
46h	ADCPPB1TRIPLO	ADC PPB1 Trip Low/Trigger Time Stamp Register	EALLOW	<a href="#">Go</a>
48h	ADCPPB2CONFIG	ADC PPB2 Config Register	EALLOW	<a href="#">Go</a>
49h	ADCPPB2STAMP	ADC PPB2 Sample Delay Time Stamp Register		<a href="#">Go</a>
4Ah	ADCPPB2OFFCAL	ADC PPB2 Offset Calibration Register	EALLOW	<a href="#">Go</a>
4Bh	ADCPPB2OFFREF	ADC PPB2 Offset Reference Register		<a href="#">Go</a>
4Ch	ADCPPB2TRIPHI	ADC PPB2 Trip High Register	EALLOW	<a href="#">Go</a>
4Eh	ADCPPB2TRIPLO	ADC PPB2 Trip Low/Trigger Time Stamp Register	EALLOW	<a href="#">Go</a>
50h	ADCPPB3CONFIG	ADC PPB3 Config Register	EALLOW	<a href="#">Go</a>
51h	ADCPPB3STAMP	ADC PPB3 Sample Delay Time Stamp Register		<a href="#">Go</a>
52h	ADCPPB3OFFCAL	ADC PPB3 Offset Calibration Register	EALLOW	<a href="#">Go</a>
53h	ADCPPB3OFFREF	ADC PPB3 Offset Reference Register		<a href="#">Go</a>
54h	ADCPPB3TRIPHI	ADC PPB3 Trip High Register	EALLOW	<a href="#">Go</a>
56h	ADCPPB3TRIPLO	ADC PPB3 Trip Low/Trigger Time Stamp Register	EALLOW	<a href="#">Go</a>
58h	ADCPPB4CONFIG	ADC PPB4 Config Register	EALLOW	<a href="#">Go</a>
59h	ADCPPB4STAMP	ADC PPB4 Sample Delay Time Stamp Register		<a href="#">Go</a>
5Ah	ADCPPB4OFFCAL	ADC PPB4 Offset Calibration Register	EALLOW	<a href="#">Go</a>
5Bh	ADCPPB4OFFREF	ADC PPB4 Offset Reference Register		<a href="#">Go</a>
5Ch	ADCPPB4TRIPHI	ADC PPB4 Trip High Register	EALLOW	<a href="#">Go</a>
5Eh	ADCPPB4TRIPLO	ADC PPB4 Trip Low/Trigger Time Stamp Register	EALLOW	<a href="#">Go</a>
6Fh	ADCINTCYCLE	ADC Early Interrupt Generation Cycle	EALLOW	<a href="#">Go</a>
70h	ADCINLTRIM1	ADC Linearity Trim 1 Register	EALLOW	<a href="#">Go</a>
72h	ADCINLTRIM2	ADC Linearity Trim 2 Register	EALLOW	<a href="#">Go</a>
74h	ADCINLTRIM3	ADC Linearity Trim 3 Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 13-12](#) shows the codes that are used for access types in this section.

**Table 13-12. ADC\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		

**Table 13-12. ADC\_REGS Access Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.



**13.16.2.1 ADCCTL1 Register (Offset = 0h) [reset = 0h]**

ADCCTL1 is shown in [Figure 13-21](#) and described in [Table 13-13](#).

Return to the [Summary Table](#).

ADC Control 1 Register

**Figure 13-21. ADCCTL1 Register**

15	14	13	12	11	10	9	8
RESERVED		ADCBSY	RESERVED	ADCBSYCHN			
R-0h		R-0h	R-0h	R-0h			
7	6	5	4	3	2	1	0
ADCPWDNZ	RESERVED				INTPULSEPOS	RESERVED	
R/W-0h	R-0h			R/W-0h		R-0h	

**Table 13-13. ADCCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0h	Reserved
13	ADCBSY	R	0h	ADC Busy. Set when ADC SOC is generated, cleared by hardware four ADC clocks after negative edge of S/H pulse. Used by the ADC state machine to determine if ADC is available to sample.  0 ADC is available to sample next channel 1 ADC is busy and cannot sample another channel Reset type: SYSRSn
12	RESERVED	R	0h	Reserved
11-8	ADCBSYCHN	R	0h	ADC Busy Channel. Set when an ADC Start of Conversion (SOC) is generated. When ADCBSY=0: holds the value of the last converted SOC When ADCBSY=1: reflects the SOC currently being processed 0h SOC0 is currently processing or was last SOC converted 1h SOC1 is currently processing or was last SOC converted 2h SOC2 is currently processing or was last SOC converted 3h SOC3 is currently processing or was last SOC converted 4h SOC4 is currently processing or was last SOC converted 5h SOC5 is currently processing or was last SOC converted 6h SOC6 is currently processing or was last SOC converted 7h SOC7 is currently processing or was last SOC converted 8h SOC8 is currently processing or was last SOC converted 9h SOC9 is currently processing or was last SOC converted Ah SOC10 is currently processing or was last SOC converted Bh SOC11 is currently processing or was last SOC converted Ch SOC12 is currently processing or was last SOC converted Dh SOC13 is currently processing or was last SOC converted Eh SOC14 is currently processing or was last SOC converted Fh SOC15 is currently processing or was last SOC converted Reset type: SYSRSn
7	ADCPWDNZ	R/W	0h	ADC Power Down (active low). This bit controls the power up and power down of all the analog circuitry inside the analog core.  0 All analog circuitry inside the core is powered down 1 All analog circuitry inside the core is powered up Reset type: SYSRSn
6-3	RESERVED	R	0h	Reserved

**Table 13-13. ADCCTL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INTPULSEPOS	R/W	0h	ADC Interrupt Pulse Position.  0 Interrupt pulse generation occurs when ADC begins conversion (at the end of the acquisition window) plus a number of SYSCLK cycles as specified in the ADCINTCYCLE.OFFSET register.  1 Interrupt pulse generation occurs at the end of the conversion, 1 cycle prior to the ADC result latching into its result register Reset type: SYSRSn
1-0	RESERVED	R	0h	Reserved

**13.16.2.2 ADCCTL2 Register (Offset = 1h) [reset = 0h]**

ADCCTL2 is shown in [Figure 13-22](#) and described in [Table 13-14](#).

Return to the [Summary Table](#).

ADC Control 2 Register

**Figure 13-22. ADCCTL2 Register**

15	14	13	12	11	10	9	8
RESERVED				RESERVED			
R-0h				R-0h			
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED		PRESCALE			
R-0h				R/W-0h			

**Table 13-14. ADCCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-8	RESERVED	R	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5-4	RESERVED	R	0h	Reserved
3-0	PRESCALE	R/W	0h	ADC Clock Prescaler. 0000 ADCCLK = Input Clock / 1.0 0001 Reserved 0010 ADCCLK = Input Clock / 2.0 0011 Reserved 0100 ADCCLK = Input Clock / 3.0 0101 Reserved 0110 ADCCLK = Input Clock / 4.0 0111 Reserved 1000 ADCCLK = Input Clock / 5.0 1001 Reserved 1010 ADCCLK = Input Clock / 6.0 1011 Reserved 1100 ADCCLK = Input Clock / 7.0 1101 Reserved 1110 ADCCLK = Input Clock / 8.0 1111 Reserved Reset type: SYSRSn

### 13.16.2.3 ADCBURSTCTL Register (Offset = 2h) [reset = 0h]

ADCBURSTCTL is shown in [Figure 13-23](#) and described in [Table 13-15](#).

Return to the [Summary Table](#).

ADC Burst Control Register

**Figure 13-23. ADCBURSTCTL Register**

15	14	13	12	11	10	9	8
BURSTEN		RESERVED			BURSTSIZE		
R/W-0h		R-0h			R/W-0h		
7	6	5	4	3	2	1	0
RESERVED		BURSTTRIGSEL					
R-0h		R/W-0h					

**Table 13-15. ADCBURSTCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	BURSTEN	R/W	0h	SOC Burst Mode Enable. This bit enables the SOC Burst Mode of operation.  0 Burst mode is disabled. 1 Burst mode is enabled. Reset type: SYSRSn
14-12	RESERVED	R	0h	Reserved
11-8	BURSTSIZE	R/W	0h	SOC Burst Size Select. This bit field determines how many SOC are converted when a burst conversion sequence is started. The first SOC converted is defined by the round robin pointer, which is advanced as each SOC is converted.  0h 1 SOC converted 1h 2 SOC converted 2h 3 SOC converted 3h 4 SOC converted 4h 5 SOC converted 5h 6 SOC converted 6h 7 SOC converted 7h 8 SOC converted 8h 9 SOC converted 9h 10 SOC converted Ah 11 SOC converted Bh 12 SOC converted Ch 13 SOC converted Dh 14 SOC converted Eh 15 SOC converted Fh 16 SOC converted Reset type: SYSRSn
7-6	RESERVED	R	0h	Reserved

**Table 13-15. ADCBURSTCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-0	BURSTTRIGSEL	R/W	0h	<p>SOC Burst Trigger Source Select. Configures which trigger will start a burst conversion sequence.</p> <p>00h BURSTTRIG0 - Software only</p> <p>01h BURSTTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h BURSTTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h BURSTTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h BURSTTRIG4 - GPIO, Input X-Bar INPUT5</p> <p>05h BURSTTRIG5 - ePWM1, ADCSOCA</p> <p>06h BURSTTRIG6 - ePWM1, ADCSOCB</p> <p>07h BURSTTRIG7 - ePWM2, ADCSOCA</p> <p>08h BURSTTRIG8 - ePWM2, ADCSOCB</p> <p>09h BURSTTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah BURSTTRIG10 - ePWM3, ADCSOCB</p> <p>0Bh BURSTTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch BURSTTRIG12 - ePWM4, ADCSOCB</p> <p>0Dh BURSTTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh BURSTTRIG14 - ePWM5, ADCSOCB</p> <p>0Fh BURSTTRIG15 - ePWM6, ADCSOCA</p> <p>10h BURSTTRIG16 - ePWM6, ADCSOCB</p> <p>11h BURSTTRIG17 - ePWM7, ADCSOCA</p> <p>12h BURSTTRIG18 - ePWM7, ADCSOCB</p> <p>13h BURSTTRIG19 - ePWM8, ADCSOCA</p> <p>14h BURSTTRIG20 - ePWM8, ADCSOCB</p> <p>15h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>

### 13.16.2.4 ADCINTFLG Register (Offset = 3h) [reset = 0h]

ADCINTFLG is shown in [Figure 13-24](#) and described in [Table 13-16](#).

Return to the [Summary Table](#).

ADC Interrupt Flag Register

**Figure 13-24. ADCINTFLG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0h				R-0h	R-0h	R-0h	R-0h

**Table 13-16. ADCINTFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R	0h	<p>ADC Interrupt 4 Flag. Reading these flags indicates if the associated ADCINT pulse was generated since the last clear.</p> <p>0 No ADC interrupt pulse generated 1 ADC interrupt pulse generated</p> <p>If the ADC interrupt is placed in continue to interrupt mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register.</p> <p>Reset type: SYSRSn</p>
2	ADCINT3	R	0h	<p>ADC Interrupt 3 Flag. Reading these flags indicates if the associated ADCINT pulse was generated since the last clear.</p> <p>0 No ADC interrupt pulse generated 1 ADC interrupt pulse generated</p> <p>If the ADC interrupt is placed in continue to interrupt mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register.</p> <p>Reset type: SYSRSn</p>
1	ADCINT2	R	0h	<p>ADC Interrupt 2 Flag. Reading these flags indicates if the associated ADCINT pulse was generated since the last clear.</p> <p>0 No ADC interrupt pulse generated 1 ADC interrupt pulse generated</p> <p>If the ADC interrupt is placed in continue to interrupt mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register.</p> <p>Reset type: SYSRSn</p>

**Table 13-16. ADCINTFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	ADCINT1	R	0h	<p>ADC Interrupt 1 Flag. Reading these flags indicates if the associated ADCINT pulse was generated since the last clear.</p> <p>0 No ADC interrupt pulse generated 1 ADC interrupt pulse generated</p> <p>If the ADC interrupt is placed in continue to interrupt mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register.</p> <p>Reset type: SYSRSn</p>

**13.16.2.5 ADCINTFLGCLR Register (Offset = 4h) [reset = 0h]**

ADCINTFLGCLR is shown in [Figure 13-25](#) and described in [Table 13-17](#).

Return to the [Summary Table](#).

ADC Interrupt Flag Clear Register

**Figure 13-25. ADCINTFLGCLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0h				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 13-17. ADCINTFLGCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R-0/W1S	0h	ADC Interrupt 4 Flag Clear. Reads return 0.  0 No action 1 Clears respective flag bit in the ADCINTFLG register. If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority but the overflow bit will not be affected (retains current state)  Boundary condition: If hardware is trying to set the bit flag while software tries to clear the bit in the same cycle, the following will take place: 1. SW has priority and will clear the flag 2. HW set will be discarded no signal will propagate to the PIE from the latch 3. Overflow flag/condition will be generated Reset type: SYSRSn
2	ADCINT3	R-0/W1S	0h	ADC Interrupt 3 Flag Clear. Reads return 0.  0 No action 1 Clears respective flag bit in the ADCINTFLG register. If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority but the overflow bit will not be affected (retains current state)  Boundary condition: If hardware is trying to set the bit flag while software tries to clear the bit in the same cycle, the following will take place: 1. SW has priority and will clear the flag 2. HW set will be discarded no signal will propagate to the PIE from the latch 3. Overflow flag/condition will be generated Reset type: SYSRSn



**Table 13-17. ADCINTFLGCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	ADCINT2	R-0/W1S	0h	<p>ADC Interrupt 2 Flag Clear. Reads return 0.</p> <p>0 No action</p> <p>1 Clears respective flag bit in the ADCINTFLG register. If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority but the overflow bit will not be affected (retains current state)</p> <p>Boundary condition: If hardware is trying to set the bit flag while software tries to clear the bit in the same cycle, the following will take place:</p> <ol style="list-style-type: none"> <li>1. SW has priority and will clear the flag</li> <li>2. HW set will be discarded</li> </ol> <p>no signal will propagate to the PIE from the latch</p> <ol style="list-style-type: none"> <li>3. Overflow flag/condition will be generated</li> </ol> <p>Reset type: SYSRSn</p>
0	ADCINT1	R-0/W1S	0h	<p>ADC Interrupt 1 Flag Clear. Reads return 0.</p> <p>0 No action</p> <p>1 Clears respective flag bit in the ADCINTFLG register. If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority but the overflow bit will not be affected (retains current state)</p> <p>Boundary condition: If hardware is trying to set the bit flag while software tries to clear the bit in the same cycle, the following will take place:</p> <ol style="list-style-type: none"> <li>1. SW has priority and will clear the flag</li> <li>2. HW set will be discarded</li> </ol> <p>no signal will propagate to the PIE from the latch</p> <ol style="list-style-type: none"> <li>3. Overflow flag/condition will be generated</li> </ol> <p>Reset type: SYSRSn</p>

### 13.16.2.6 ADCINTOVF Register (Offset = 5h) [reset = 0h]

ADCINTOVF is shown in [Figure 13-26](#) and described in [Table 13-18](#).

Return to the [Summary Table](#).

ADC Interrupt Overflow Register

**Figure 13-26. ADCINTOVF Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0h				R-0h	R-0h	R-0h	R-0h

**Table 13-18. ADCINTOVF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R	0h	ADC Interrupt 4 Overflow Flags  Indicates if an overflow occurred when generating ADCINT pulses. If the respective ADCINTFLG bit is set and a selected additional EOC trigger is generated, then an overflow condition occurs.  0 No ADC interrupt overflow event detected. 1 ADC Interrupt overflow event detected.  The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection. Reset type: SYSRSn
2	ADCINT3	R	0h	ADC Interrupt 3 Overflow Flags  Indicates if an overflow occurred when generating ADCINT pulses. If the respective ADCINTFLG bit is set and a selected additional EOC trigger is generated, then an overflow condition occurs.  0 No ADC interrupt overflow event detected. 1 ADC Interrupt overflow event detected.  The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection. Reset type: SYSRSn
1	ADCINT2	R	0h	ADC Interrupt 2 Overflow Flags  Indicates if an overflow occurred when generating ADCINT pulses. If the respective ADCINTFLG bit is set and a selected additional EOC trigger is generated, then an overflow condition occurs.  0 No ADC interrupt overflow event detected. 1 ADC Interrupt overflow event detected.  The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection. Reset type: SYSRSn

**Table 13-18. ADCINTOVF Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	ADCINT1	R	0h	<p>ADC Interrupt 1 Overflow Flags</p> <p>Indicates if an overflow occurred when generating ADCINT pulses. If the respective ADCINTFLG bit is set and a selected additional EOC trigger is generated, then an overflow condition occurs.</p> <p>0 No ADC interrupt overflow event detected. 1 ADC Interrupt overflow event detected.</p> <p>The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection.</p> <p>Reset type: SYSRSn</p>

**13.16.2.7 ADCINTOVFCLR Register (Offset = 6h) [reset = 0h]**

 ADCINTOVFCLR is shown in [Figure 13-27](#) and described in [Table 13-19](#).

 Return to the [Summary Table](#).

ADC Interrupt Overflow Clear Register

**Figure 13-27. ADCINTOVFCLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0h				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 13-19. ADCINTOVFCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R-0/W1S	0h	ADC Interrupt 4 Overflow Clear Bits  0 No action. 1 Clears the respective overflow bit in the ADCINTOVF register. If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCINTOVF register, then hardware has priority and the ADCINTOVF bit will be set. Reset type: SYSRSn
2	ADCINT3	R-0/W1S	0h	ADC Interrupt 3 Overflow Clear Bits  0 No action. 1 Clears the respective overflow bit in the ADCINTOVF register. If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCINTOVF register, then hardware has priority and the ADCINTOVF bit will be set. Reset type: SYSRSn
1	ADCINT2	R-0/W1S	0h	ADC Interrupt 2 Overflow Clear Bits  0 No action. 1 Clears the respective overflow bit in the ADCINTOVF register. If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCINTOVF register, then hardware has priority and the ADCINTOVF bit will be set. Reset type: SYSRSn
0	ADCINT1	R-0/W1S	0h	ADC Interrupt 1 Overflow Clear Bits  0 No action. 1 Clears the respective overflow bit in the ADCINTOVF register. If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCINTOVF register, then hardware has priority and the ADCINTOVF bit will be set. Reset type: SYSRSn

### 13.16.2.8 ADCINTSEL1N2 Register (Offset = 7h) [reset = 0h]

ADCINTSEL1N2 is shown in [Figure 13-28](#) and described in [Table 13-20](#).

Return to the [Summary Table](#).

ADC Interrupt 1 and 2 Selection Register

**Figure 13-28. ADCINTSEL1N2 Register**

15	14	13	12	11	10	9	8
RESERVED	INT2CONT	INT2E	RESERVED	INT2SEL			
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h			
7	6	5	4	3	2	1	0
RESERVED	INT1CONT	INT1E	RESERVED	INT1SEL			
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h			

**Table 13-20. ADCINTSEL1N2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	INT2CONT	R/W	0h	ADCINT2 Continue to Interrupt Mode 0 No further ADCINT2 pulses are generated until ADCINT2 flag (in ADCINTFLG register) is cleared by user. 1 ADCINT2 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not. Reset type: SYSRSn
13	INT2E	R/W	0h	ADCINT2 Interrupt Enable 0 ADCINT2 is disabled 1 ADCINT2 is enabled Reset type: SYSRSn
12	RESERVED	R	0h	Reserved
11-8	INT2SEL	R/W	0h	ADCINT2 EOC Source Select 0h EOC0 is trigger for ADCINT2 1h EOC1 is trigger for ADCINT2 2h EOC2 is trigger for ADCINT2 3h EOC3 is trigger for ADCINT2 4h EOC4 is trigger for ADCINT2 5h EOC5 is trigger for ADCINT2 6h EOC6 is trigger for ADCINT2 7h EOC7 is trigger for ADCINT2 8h EOC8 is trigger for ADCINT2 9h EOC9 is trigger for ADCINT2 Ah EOC10 is trigger for ADCINT2 Bh EOC11 is trigger for ADCINT2 Ch EOC12 is trigger for ADCINT2 Dh EOC13 is trigger for ADCINT2 Eh EOC14 is trigger for ADCINT2 Fh EOC15 is trigger for ADCINT2 Reset type: SYSRSn
7	RESERVED	R	0h	Reserved

**Table 13-20. ADCINTSEL1N2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	INT1CONT	R/W	0h	ADCINT1 Continue to Interrupt Mode 0 No further ADCINT1 pulses are generated until ADCINT1 flag (in ADCINTFLG register) is cleared by user. 1 ADCINT1 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not. Reset type: SYSRSn
5	INT1E	R/W	0h	ADCINT1 Interrupt Enable 0 ADCINT1 is disabled 1 ADCINT1 is enabled Reset type: SYSRSn
4	RESERVED	R	0h	Reserved
3-0	INT1SEL	R/W	0h	ADCINT1 EOC Source Select 0h EOC0 is trigger for ADCINT1 1h EOC1 is trigger for ADCINT1 2h EOC2 is trigger for ADCINT1 3h EOC3 is trigger for ADCINT1 4h EOC4 is trigger for ADCINT1 5h EOC5 is trigger for ADCINT1 6h EOC6 is trigger for ADCINT1 7h EOC7 is trigger for ADCINT1 8h EOC8 is trigger for ADCINT1 9h EOC9 is trigger for ADCINT1 Ah EOC10 is trigger for ADCINT1 Bh EOC11 is trigger for ADCINT1 Ch EOC12 is trigger for ADCINT1 Dh EOC13 is trigger for ADCINT1 Eh EOC14 is trigger for ADCINT1 Fh EOC15 is trigger for ADCINT1 Reset type: SYSRSn

### 13.16.2.9 ADCINTSEL3N4 Register (Offset = 8h) [reset = 0h]

ADCINTSEL3N4 is shown in [Figure 13-29](#) and described in [Table 13-21](#).

Return to the [Summary Table](#).

ADC Interrupt 3 and 4 Selection Register

**Figure 13-29. ADCINTSEL3N4 Register**

15	14	13	12	11	10	9	8
RESERVED	INT4CONT	INT4E	RESERVED	INT4SEL			
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h			
7	6	5	4	3	2	1	0
RESERVED	INT3CONT	INT3E	RESERVED	INT3SEL			
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h			

**Table 13-21. ADCINTSEL3N4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	INT4CONT	R/W	0h	ADCINT4 Continue to Interrupt Mode 0 No further ADCINT4 pulses are generated until ADCINT4 flag (in ADCINTFLG register) is cleared by user. 1 ADCINT4 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not. Reset type: SYSRSn
13	INT4E	R/W	0h	ADCINT4 Interrupt Enable 0 ADCINT4 is disabled 1 ADCINT4 is enabled Reset type: SYSRSn
12	RESERVED	R	0h	Reserved
11-8	INT4SEL	R/W	0h	ADCINT4 EOC Source Select 0h EOC0 is trigger for ADCINT4 1h EOC1 is trigger for ADCINT4 2h EOC2 is trigger for ADCINT4 3h EOC3 is trigger for ADCINT4 4h EOC4 is trigger for ADCINT4 5h EOC5 is trigger for ADCINT4 6h EOC6 is trigger for ADCINT4 7h EOC7 is trigger for ADCINT4 8h EOC8 is trigger for ADCINT4 9h EOC9 is trigger for ADCINT4 Ah EOC10 is trigger for ADCINT4 Bh EOC11 is trigger for ADCINT4 Ch EOC12 is trigger for ADCINT4 Dh EOC13 is trigger for ADCINT4 Eh EOC14 is trigger for ADCINT4 Fh EOC15 is trigger for ADCINT4 Reset type: SYSRSn
7	RESERVED	R	0h	Reserved

**Table 13-21. ADCINTSEL3N4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	INT3CONT	R/W	0h	ADCINT3 Continue to Interrupt Mode 0 No further ADCINT3 pulses are generated until ADCINT3 flag (in ADCINTFLG register) is cleared by user. 1 ADCINT3 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not. Reset type: SYSRSn
5	INT3E	R/W	0h	ADCINT3 Interrupt Enable 0 ADCINT3 is disabled 1 ADCINT3 is enabled Reset type: SYSRSn
4	RESERVED	R	0h	Reserved
3-0	INT3SEL	R/W	0h	ADCINT3 EOC Source Select 0h EOC0 is trigger for ADCINT3 1h EOC1 is trigger for ADCINT3 2h EOC2 is trigger for ADCINT3 3h EOC3 is trigger for ADCINT3 4h EOC4 is trigger for ADCINT3 5h EOC5 is trigger for ADCINT3 6h EOC6 is trigger for ADCINT3 7h EOC7 is trigger for ADCINT3 8h EOC8 is trigger for ADCINT3 9h EOC9 is trigger for ADCINT3 Ah EOC10 is trigger for ADCINT3 Bh EOC11 is trigger for ADCINT3 Ch EOC12 is trigger for ADCINT3 Dh EOC13 is trigger for ADCINT3 Eh EOC14 is trigger for ADCINT3 Fh EOC15 is trigger for ADCINT3 Reset type: SYSRSn



**13.16.2.10 ADCSOCPRCTL Register (Offset = 9h) [reset = 200h]**

ADCSOCPRCTL is shown in [Figure 13-30](#) and described in [Table 13-22](#).

Return to the [Summary Table](#).

ADC SOC Priority Control Register

**Figure 13-30. ADCSOCPRCTL Register**

15	14	13	12	11	10	9	8
RESERVED						RRPOINTER	
R-0h						R-10h	
7	6	5	4	3	2	1	0
RRPOINTER			SOCPRIORITY				
R-10h			R/W-0h				

**Table 13-22. ADCSOCPRCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved

**Table 13-22. ADCSOCPRICTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-5	RRPOINTER	R	10h	<p>Round Robin Pointer. Holds the value of the last converted round robin SOCx to be used by the round robin scheme to determine order of conversions.</p> <p>00h SOC0 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>01h SOC1 was last round robin SOC to convert, SOC2 is highest round robin priority.</p> <p>02h SOC2 was last round robin SOC to convert, SOC3 is highest round robin priority.</p> <p>03h SOC3 was last round robin SOC to convert, SOC4 is highest round robin priority.</p> <p>04h SOC4 was last round robin SOC to convert, SOC5 is highest round robin priority.</p> <p>05h SOC5 was last round robin SOC to convert, SOC6 is highest round robin priority.</p> <p>06h SOC6 was last round robin SOC to convert, SOC7 is highest round robin priority.</p> <p>07h SOC7 was last round robin SOC to convert, SOC8 is highest round robin priority.</p> <p>08h SOC8 was last round robin SOC to convert, SOC9 is highest round robin priority.</p> <p>09h SOC9 was last round robin SOC to convert, SOC10 is highest round robin priority.</p> <p>0Ah SOC10 was last round robin SOC to convert, SOC11 is highest round robin priority.</p> <p>0Bh SOC11 was last round robin SOC to convert, SOC12 is highest round robin priority.</p> <p>0Ch SOC12 was last round robin SOC to convert, SOC13 is highest round robin priority.</p> <p>0Dh SOC13 was last round robin SOC to convert, SOC14 is highest round robin priority.</p> <p>0Eh SOC14 was last round robin SOC to convert, SOC15 is highest round robin priority.</p> <p>0Fh SOC15 was last round robin SOC to convert, SOC0 is highest round robin priority.</p> <p>10h Reset value to indicate no SOC has been converted. SOC0 is highest round robin priority. Set to this value when the device is reset, when the ADCCTL1.RESET bit is set, or when the ADCSOPRICTL register is written. In the latter case, if a conversion is currently in progress, it will complete and then the new priority will take effect.</p> <p>Others Invalid value.</p> <p>Reset type: SYSRSn</p>

**Table 13-22. ADCSOCPRCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	SOC PRIORITY	R/W	0h	<p>SOC Priority</p> <p>Determines the cutoff point for priority mode and round robin arbitration for SOCx</p> <p>00h SOC priority is handled in round robin mode for all channels.</p> <p>01h SOC0 is high priority, rest of channels are in round robin mode.</p> <p>02h SOC0-SOC1 are high priority, SOC2-SOC15 are in round robin mode.</p> <p>03h SOC0-SOC2 are high priority, SOC3-SOC15 are in round robin mode.</p> <p>04h SOC0-SOC3 are high priority, SOC4-SOC15 are in round robin mode.</p> <p>05h SOC0-SOC4 are high priority, SOC5-SOC15 are in round robin mode.</p> <p>06h SOC0-SOC5 are high priority, SOC6-SOC15 are in round robin mode.</p> <p>07h SOC0-SOC6 are high priority, SOC7-SOC15 are in round robin mode.</p> <p>08h SOC0-SOC7 are high priority, SOC8-SOC15 are in round robin mode.</p> <p>09h SOC0-SOC8 are high priority, SOC9-SOC15 are in round robin mode.</p> <p>0Ah SOC0-SOC9 are high priority, SOC10-SOC15 are in round robin mode.</p> <p>0Bh SOC0-SOC10 are high priority, SOC11-SOC15 are in round robin mode.</p> <p>0Ch SOC0-SOC11 are high priority, SOC12-SOC15 are in round robin mode.</p> <p>0Dh SOC0-SOC12 are high priority, SOC13-SOC15 are in round robin mode.</p> <p>0Eh SOC0-SOC13 are high priority, SOC14-SOC15 are in round robin mode.</p> <p>0Fh SOC0-SOC14 are high priority, SOC15 is in round robin mode.</p> <p>10h All SOCx are in high priority mode, arbitrated by SOC number.</p> <p>Others Invalid selection.</p> <p>Reset type: SYSRSn</p>

**13.16.2.11 ADCINTSOCSEL1 Register (Offset = Ah) [reset = 0h]**

 ADCINTSOCSEL1 is shown in [Figure 13-31](#) and described in [Table 13-23](#).

 Return to the [Summary Table](#).

ADC Interrupt SOC Selection 1 Register

**Figure 13-31. ADCINTSOCSEL1 Register**

15	14	13	12	11	10	9	8
SOC7		SOC6		SOC5		SOC4	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
SOC3		SOC2		SOC1		SOC0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 13-23. ADCINTSOCSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	SOC7	R/W	0h	SOC7 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC7. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC7. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC7. 10 ADCINT2 will trigger SOC7. 11 Invalid selection. Reset type: SYSRSn
13-12	SOC6	R/W	0h	SOC6 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC6. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC6. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC6. 10 ADCINT2 will trigger SOC6. 11 Invalid selection. Reset type: SYSRSn
11-10	SOC5	R/W	0h	SOC5 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC5. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC5. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC5. 10 ADCINT2 will trigger SOC5. 11 Invalid selection. Reset type: SYSRSn
9-8	SOC4	R/W	0h	SOC4 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC4. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC4. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC4. 10 ADCINT2 will trigger SOC4. 11 Invalid selection. Reset type: SYSRSn

**Table 13-23. ADCINTSOCSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	SOC3	R/W	0h	<p>SOC3 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC3. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC3. TRIGSEL field alone determines SOC0 trigger.</p> <p>01 ADCINT1 will trigger SOC3.</p> <p>10 ADCINT2 will trigger SOC3.</p> <p>11 Invalid selection.</p> <p>Reset type: SYSRSn</p>
5-4	SOC2	R/W	0h	<p>SOC2 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC2. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC2. TRIGSEL field alone determines SOC0 trigger.</p> <p>01 ADCINT1 will trigger SOC2.</p> <p>10 ADCINT2 will trigger SOC2.</p> <p>11 Invalid selection.</p> <p>Reset type: SYSRSn</p>
3-2	SOC1	R/W	0h	<p>SOC1 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC1. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC1. TRIGSEL field alone determines SOC0 trigger.</p> <p>01 ADCINT1 will trigger SOC1.</p> <p>10 ADCINT2 will trigger SOC1.</p> <p>11 Invalid selection.</p> <p>Reset type: SYSRSn</p>
1-0	SOC0	R/W	0h	<p>SOC0 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC0. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC0. TRIGSEL field alone determines SOC0 trigger.</p> <p>01 ADCINT1 will trigger SOC0.</p> <p>10 ADCINT2 will trigger SOC0.</p> <p>11 Invalid selection.</p> <p>Reset type: SYSRSn</p>

### 13.16.2.12 ADCINTSOCSEL2 Register (Offset = Bh) [reset = 0h]

ADCINTSOCSEL2 is shown in [Figure 13-32](#) and described in [Table 13-24](#).

Return to the [Summary Table](#).

ADC Interrupt SOC Selection 2 Register

**Figure 13-32. ADCINTSOCSEL2 Register**

15	14	13	12	11	10	9	8
SOC15		SOC14		SOC13		SOC12	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
SOC11		SOC10		SOC9		SOC8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 13-24. ADCINTSOCSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	SOC15	R/W	0h	SOC15 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC15. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC15. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC15. 10 ADCINT2 will trigger SOC15. 11 Invalid selection. Reset type: SYSRSn
13-12	SOC14	R/W	0h	SOC14 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC14. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC14. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC14. 10 ADCINT2 will trigger SOC14. 11 Invalid selection. Reset type: SYSRSn
11-10	SOC13	R/W	0h	SOC13 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC13. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC13. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC13. 10 ADCINT2 will trigger SOC13. 11 Invalid selection. Reset type: SYSRSn
9-8	SOC12	R/W	0h	SOC12 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC12. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC12. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC12. 10 ADCINT2 will trigger SOC12. 11 Invalid selection. Reset type: SYSRSn

**Table 13-24. ADCINTSOCSEL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	SOC11	R/W	0h	<p>SOC11 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC11. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC11. TRIGSEL field alone determines SOC0 trigger.</p> <p>01 ADCINT1 will trigger SOC11.</p> <p>10 ADCINT2 will trigger SOC11.</p> <p>11 Invalid selection.</p> <p>Reset type: SYSRSn</p>
5-4	SOC10	R/W	0h	<p>SOC10 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC10. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC10. TRIGSEL field alone determines SOC0 trigger.</p> <p>01 ADCINT1 will trigger SOC10.</p> <p>10 ADCINT2 will trigger SOC10.</p> <p>11 Invalid selection.</p> <p>Reset type: SYSRSn</p>
3-2	SOC9	R/W	0h	<p>SOC9 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC9. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC9. TRIGSEL field alone determines SOC0 trigger.</p> <p>01 ADCINT1 will trigger SOC9.</p> <p>10 ADCINT2 will trigger SOC9.</p> <p>11 Invalid selection.</p> <p>Reset type: SYSRSn</p>
1-0	SOC8	R/W	0h	<p>SOC8 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC8. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC8. TRIGSEL field alone determines SOC0 trigger.</p> <p>01 ADCINT1 will trigger SOC8.</p> <p>10 ADCINT2 will trigger SOC8.</p> <p>11 Invalid selection.</p> <p>Reset type: SYSRSn</p>

### 13.16.2.13 ADCSOCFLG1 Register (Offset = Ch) [reset = 0h]

ADCSOCFLG1 is shown in [Figure 13-33](#) and described in [Table 13-25](#).

Return to the [Summary Table](#).

ADC SOC Flag 1 Register

**Figure 13-33. ADCSOCFLG1 Register**

15		14		13		12		11		10		9		8	
SOC15		SOC14		SOC13		SOC12		SOC11		SOC10		SOC9		SOC8	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h	
7		6		5		4		3		2		1		0	
SOC7		SOC6		SOC5		SOC4		SOC3		SOC2		SOC1		SOC0	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h	

**Table 13-25. ADCSOCFLG1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOC15	R	0h	<p>SOC15 Start of Conversion Flag. Indicates the state of SOC15 conversions.</p> <p>0 No sample pending for SOC15. 1 Trigger has been received and sample is pending for SOC15.</p> <p>This bit will be automatically cleared when the SOC15 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
14	SOC14	R	0h	<p>SOC14 Start of Conversion Flag. Indicates the state of SOC14 conversions.</p> <p>0 No sample pending for SOC14. 1 Trigger has been received and sample is pending for SOC14.</p> <p>This bit will be automatically cleared when the SOC14 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
13	SOC13	R	0h	<p>SOC13 Start of Conversion Flag. Indicates the state of SOC13 conversions.</p> <p>0 No sample pending for SOC13. 1 Trigger has been received and sample is pending for SOC13.</p> <p>This bit will be automatically cleared when the SOC13 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>



**Table 13-25. ADCSOCFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	SOC12	R	0h	<p>SOC12 Start of Conversion Flag. Indicates the state of SOC12 conversions.</p> <p>0 No sample pending for SOC12. 1 Trigger has been received and sample is pending for SOC12.</p> <p>This bit will be automatically cleared when the SOC12 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
11	SOC11	R	0h	<p>SOC11 Start of Conversion Flag. Indicates the state of SOC11 conversions.</p> <p>0 No sample pending for SOC11. 1 Trigger has been received and sample is pending for SOC11.</p> <p>This bit will be automatically cleared when the SOC11 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
10	SOC10	R	0h	<p>SOC10 Start of Conversion Flag. Indicates the state of SOC10 conversions.</p> <p>0 No sample pending for SOC10. 1 Trigger has been received and sample is pending for SOC10.</p> <p>This bit will be automatically cleared when the SOC10 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
9	SOC9	R	0h	<p>SOC9 Start of Conversion Flag. Indicates the state of SOC9 conversions.</p> <p>0 No sample pending for SOC9. 1 Trigger has been received and sample is pending for SOC9.</p> <p>This bit will be automatically cleared when the SOC9 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 13-25. ADCSOCFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	SOC8	R	0h	<p>SOC8 Start of Conversion Flag. Indicates the state of SOC8 conversions.</p> <p>0 No sample pending for SOC8. 1 Trigger has been received and sample is pending for SOC8.</p> <p>This bit will be automatically cleared when the SOC8 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
7	SOC7	R	0h	<p>SOC7 Start of Conversion Flag. Indicates the state of SOC7 conversions.</p> <p>0 No sample pending for SOC7. 1 Trigger has been received and sample is pending for SOC7.</p> <p>This bit will be automatically cleared when the SOC7 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
6	SOC6	R	0h	<p>SOC6 Start of Conversion Flag. Indicates the state of SOC6 conversions.</p> <p>0 No sample pending for SOC6. 1 Trigger has been received and sample is pending for SOC6.</p> <p>This bit will be automatically cleared when the SOC6 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
5	SOC5	R	0h	<p>SOC5 Start of Conversion Flag. Indicates the state of SOC5 conversions.</p> <p>0 No sample pending for SOC5. 1 Trigger has been received and sample is pending for SOC5.</p> <p>This bit will be automatically cleared when the SOC5 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 13-25. ADCSOCFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	SOC4	R	0h	<p>SOC4 Start of Conversion Flag. Indicates the state of SOC4 conversions.</p> <p>0 No sample pending for SOC4. 1 Trigger has been received and sample is pending for SOC4.</p> <p>This bit will be automatically cleared when the SOC4 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
3	SOC3	R	0h	<p>SOC3 Start of Conversion Flag. Indicates the state of SOC3 conversions.</p> <p>0 No sample pending for SOC3. 1 Trigger has been received and sample is pending for SOC3.</p> <p>This bit will be automatically cleared when the SOC3 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
2	SOC2	R	0h	<p>SOC2 Start of Conversion Flag. Indicates the state of SOC2 conversions.</p> <p>0 No sample pending for SOC2. 1 Trigger has been received and sample is pending for SOC2.</p> <p>This bit will be automatically cleared when the SOC2 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
1	SOC1	R	0h	<p>SOC1 Start of Conversion Flag. Indicates the state of SOC1 conversions.</p> <p>0 No sample pending for SOC1. 1 Trigger has been received and sample is pending for SOC1.</p> <p>This bit will be automatically cleared when the SOC1 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 13-25. ADCSOCFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	SOC0	R	0h	<p>SOC0 Start of Conversion Flag. Indicates the state of SOC0 conversions.</p> <p>0 No sample pending for SOC0. 1 Trigger has been received and sample is pending for SOC0.</p> <p>This bit will be automatically cleared when the SOC0 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

### 13.16.2.14 ADCSOCFRC1 Register (Offset = Dh) [reset = 0h]

ADCSOCFRC1 is shown in [Figure 13-34](#) and described in [Table 13-26](#).

Return to the [Summary Table](#).

ADC SOC Force 1 Register

**Figure 13-34. ADCSOCFRC1 Register**

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 13-26. ADCSOCFRC1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOC15	R-0/W1S	0h	<p>SOC15 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC15 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC15 flag bit to 1. This will cause a conversion to start once priority is given to SOC15.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC15 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
14	SOC14	R-0/W1S	0h	<p>SOC14 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC14 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC14 flag bit to 1. This will cause a conversion to start once priority is given to SOC14.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC14 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 13-26. ADCSOCFRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	SOC13	R-0/W1S	0h	<p>SOC13 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC13 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC13 flag bit to 1. This will cause a conversion to start once priority is given to SOC13.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC13 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
12	SOC12	R-0/W1S	0h	<p>SOC12 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC12 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC12 flag bit to 1. This will cause a conversion to start once priority is given to SOC12.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC12 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
11	SOC11	R-0/W1S	0h	<p>SOC11 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC11 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC11 flag bit to 1. This will cause a conversion to start once priority is given to SOC11.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC11 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
10	SOC10	R-0/W1S	0h	<p>SOC10 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC10 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC10 flag bit to 1. This will cause a conversion to start once priority is given to SOC10.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC10 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 13-26. ADCSOCFRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	SOC9	R-0/W1S	0h	<p>SOC9 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC9 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC9 flag bit to 1. This will cause a conversion to start once priority is given to SOC9.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC9 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
8	SOC8	R-0/W1S	0h	<p>SOC8 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC8 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC8 flag bit to 1. This will cause a conversion to start once priority is given to SOC8.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC8 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
7	SOC7	R-0/W1S	0h	<p>SOC7 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC7 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC7 flag bit to 1. This will cause a conversion to start once priority is given to SOC7.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC7 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
6	SOC6	R-0/W1S	0h	<p>SOC6 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC6 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC6 flag bit to 1. This will cause a conversion to start once priority is given to SOC6.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC6 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 13-26. ADCSOCFRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	SOC5	R-0/W1S	0h	<p>SOC5 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC5 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC5 flag bit to 1. This will cause a conversion to start once priority is given to SOC5.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC5 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
4	SOC4	R-0/W1S	0h	<p>SOC4 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC4 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC4 flag bit to 1. This will cause a conversion to start once priority is given to SOC4.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC4 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
3	SOC3	R-0/W1S	0h	<p>SOC3 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC3 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC3 flag bit to 1. This will cause a conversion to start once priority is given to SOC3.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC3 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
2	SOC2	R-0/W1S	0h	<p>SOC2 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC2 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC2 flag bit to 1. This will cause a conversion to start once priority is given to SOC2.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC2 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>



**Table 13-26. ADCSOCFRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	SOC1	R-0/W1S	0h	<p>SOC1 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC1 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action. 1 Force SOC1 flag bit to 1. This will cause a conversion to start once priority is given to SOC1.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC1 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
0	SOC0	R-0/W1S	0h	<p>SOC0 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC0 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action. 1 Force SOC0 flag bit to 1. This will cause a conversion to start once priority is given to SOC0.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC0 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**13.16.2.15 ADCSOCOVF1 Register (Offset = Eh) [reset = 0h]**

ADCSOCOVF1 is shown in [Figure 13-35](#) and described in [Table 13-27](#).

Return to the [Summary Table](#).

ADC SOC Overflow 1 Register

**Figure 13-35. ADCSOCOVF1 Register**

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 13-27. ADCSOCOVF1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOC15	R	0h	<p>SOC15 Start of Conversion Overflow Flag. Indicates an SOC15 event was generated in hardware while an existing SOC15 event was already pending.</p> <p>0 No SOC15 event overflow. 1 SOC15 event overflow.</p> <p>An overflow condition does not stop SOC15 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
14	SOC14	R	0h	<p>SOC14 Start of Conversion Overflow Flag. Indicates an SOC14 event was generated in hardware while an existing SOC14 event was already pending.</p> <p>0 No SOC14 event overflow. 1 SOC14 event overflow.</p> <p>An overflow condition does not stop SOC14 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
13	SOC13	R	0h	<p>SOC13 Start of Conversion Overflow Flag. Indicates an SOC13 event was generated in hardware while an existing SOC13 event was already pending.</p> <p>0 No SOC13 event overflow. 1 SOC13 event overflow.</p> <p>An overflow condition does not stop SOC13 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>

**Table 13-27. ADCSOCOVF1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	SOC12	R	0h	<p>SOC12 Start of Conversion Overflow Flag. Indicates an SOC12 event was generated in hardware while an existing SOC12 event was already pending.</p> <p>0 No SOC12 event overflow. 1 SOC12 event overflow.</p> <p>An overflow condition does not stop SOC12 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
11	SOC11	R	0h	<p>SOC11 Start of Conversion Overflow Flag. Indicates an SOC11 event was generated in hardware while an existing SOC11 event was already pending.</p> <p>0 No SOC11 event overflow. 1 SOC11 event overflow.</p> <p>An overflow condition does not stop SOC11 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
10	SOC10	R	0h	<p>SOC10 Start of Conversion Overflow Flag. Indicates an SOC10 event was generated in hardware while an existing SOC10 event was already pending.</p> <p>0 No SOC10 event overflow. 1 SOC10 event overflow.</p> <p>An overflow condition does not stop SOC10 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
9	SOC9	R	0h	<p>SOC9 Start of Conversion Overflow Flag. Indicates an SOC9 event was generated in hardware while an existing SOC9 event was already pending.</p> <p>0 No SOC9 event overflow. 1 SOC9 event overflow.</p> <p>An overflow condition does not stop SOC9 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
8	SOC8	R	0h	<p>SOC8 Start of Conversion Overflow Flag. Indicates an SOC8 event was generated in hardware while an existing SOC8 event was already pending.</p> <p>0 No SOC8 event overflow. 1 SOC8 event overflow.</p> <p>An overflow condition does not stop SOC8 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>

**Table 13-27. ADCSOCOVF1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	SOC7	R	0h	<p>SOC7 Start of Conversion Overflow Flag. Indicates an SOC7 event was generated in hardware while an existing SOC7 event was already pending.</p> <p>0 No SOC7 event overflow. 1 SOC7 event overflow.</p> <p>An overflow condition does not stop SOC7 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
6	SOC6	R	0h	<p>SOC6 Start of Conversion Overflow Flag. Indicates an SOC6 event was generated in hardware while an existing SOC6 event was already pending.</p> <p>0 No SOC6 event overflow. 1 SOC6 event overflow.</p> <p>An overflow condition does not stop SOC6 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
5	SOC5	R	0h	<p>SOC5 Start of Conversion Overflow Flag. Indicates an SOC5 event was generated in hardware while an existing SOC5 event was already pending.</p> <p>0 No SOC5 event overflow. 1 SOC5 event overflow.</p> <p>An overflow condition does not stop SOC5 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
4	SOC4	R	0h	<p>SOC4 Start of Conversion Overflow Flag. Indicates an SOC4 event was generated in hardware while an existing SOC4 event was already pending.</p> <p>0 No SOC4 event overflow. 1 SOC4 event overflow.</p> <p>An overflow condition does not stop SOC4 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
3	SOC3	R	0h	<p>SOC3 Start of Conversion Overflow Flag. Indicates an SOC3 event was generated in hardware while an existing SOC3 event was already pending.</p> <p>0 No SOC3 event overflow. 1 SOC3 event overflow.</p> <p>An overflow condition does not stop SOC3 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>

**Table 13-27. ADCSOCOVF1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	SOC2	R	0h	<p>SOC2 Start of Conversion Overflow Flag. Indicates an SOC2 event was generated in hardware while an existing SOC2 event was already pending.</p> <p>0 No SOC2 event overflow. 1 SOC2 event overflow.</p> <p>An overflow condition does not stop SOC2 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
1	SOC1	R	0h	<p>SOC1 Start of Conversion Overflow Flag. Indicates an SOC1 event was generated in hardware while an existing SOC1 event was already pending.</p> <p>0 No SOC1 event overflow. 1 SOC1 event overflow.</p> <p>An overflow condition does not stop SOC1 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
0	SOC0	R	0h	<p>SOC0 Start of Conversion Overflow Flag. Indicates an SOC0 event was generated in hardware while an existing SOC0 event was already pending.</p> <p>0 No SOC0 event overflow. 1 SOC0 event overflow.</p> <p>An overflow condition does not stop SOC0 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>

### 13.16.2.16 ADCSOCOVFCLR1 Register (Offset = Fh) [reset = 0h]

ADCSOCOVFCLR1 is shown in [Figure 13-36](#) and described in [Table 13-28](#).

Return to the [Summary Table](#).

ADC SOC Overflow Clear 1 Register

**Figure 13-36. ADCSOCOVFCLR1 Register**

15		14		13		12		11		10		9		8	
SOC15		SOC14		SOC13		SOC12		SOC11		SOC10		SOC9		SOC8	
R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h	
7		6		5		4		3		2		1		0	
SOC7		SOC6		SOC5		SOC4		SOC3		SOC2		SOC1		SOC0	
R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h	

**Table 13-28. ADCSOCOVFCLR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOC15	R-0/W1S	0h	<p>SOC15 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC15 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC15 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
14	SOC14	R-0/W1S	0h	<p>SOC14 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC14 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC14 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
13	SOC13	R-0/W1S	0h	<p>SOC13 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC13 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC13 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
12	SOC12	R-0/W1S	0h	<p>SOC12 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC12 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC12 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>

**Table 13-28. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	SOC11	R-0/W1S	0h	<p>SOC11 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC11 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC11 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
10	SOC10	R-0/W1S	0h	<p>SOC10 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC10 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC10 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
9	SOC9	R-0/W1S	0h	<p>SOC9 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC9 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC9 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
8	SOC8	R-0/W1S	0h	<p>SOC8 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC8 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC8 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
7	SOC7	R-0/W1S	0h	<p>SOC7 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC7 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC7 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>

**Table 13-28. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	SOC6	R-0/W1S	0h	<p>SOC6 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC6 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC6 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
5	SOC5	R-0/W1S	0h	<p>SOC5 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC5 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC5 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
4	SOC4	R-0/W1S	0h	<p>SOC4 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC4 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC4 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
3	SOC3	R-0/W1S	0h	<p>SOC3 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC3 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC3 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
2	SOC2	R-0/W1S	0h	<p>SOC2 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC2 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC2 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>



**Table 13-28. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	SOC1	R-0/W1S	0h	<p>SOC1 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC1 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC1 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
0	SOC0	R-0/W1S	0h	<p>SOC0 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC0 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC0 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>

**13.16.2.17 ADCSOC0CTL Register (Offset = 10h) [reset = 0h]**

ADCSOC0CTL is shown in [Figure 13-37](#) and described in [Table 13-29](#).

Return to the [Summary Table](#).

ADC SOC0 Control Register

**Figure 13-37. ADCSOC0CTL Register**

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

**Table 13-29. ADCSOC0CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC0 Trigger Source Select. Along with the SOC0 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC0 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCA 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCA 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCA 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCA 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCA 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCA 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCA 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCA 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-29. ADCSOC0CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC0 Channel Select. Selects the channel to be converted when SOC0 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC0 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.18 ADCSOC1CTL Register (Offset = 12h) [reset = 0h]**

ADCSOC1CTL is shown in [Figure 13-38](#) and described in [Table 13-30](#).

Return to the [Summary Table](#).

ADC SOC1 Control Register

**Figure 13-38. ADCSOC1CTL Register**

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

**Table 13-30. ADCSOC1CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC1 Trigger Source Select. Along with the SOC1 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC1 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCA 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCA 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCA 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCA 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCA 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCA 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCA 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCA 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-30. ADCSOC1CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC1 Channel Select. Selects the channel to be converted when SOC1 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC1 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.19 ADCSOC2CTL Register (Offset = 14h) [reset = 0h]**

ADCSOC2CTL is shown in [Figure 13-39](#) and described in [Table 13-31](#).

Return to the [Summary Table](#).

ADC SOC2 Control Register

**Figure 13-39. ADCSOC2CTL Register**

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

**Table 13-31. ADCSOC2CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC2 Trigger Source Select. Along with the SOC2 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC2 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-31. ADCSOC2CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC2 Channel Select. Selects the channel to be converted when SOC2 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC2 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.20 ADCSOC3CTL Register (Offset = 16h) [reset = 0h]**

 ADCSOC3CTL is shown in [Figure 13-40](#) and described in [Table 13-32](#).

 Return to the [Summary Table](#).

ADC SOC3 Control Register

**Figure 13-40. ADCSOC3CTL Register**

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

**Table 13-32. ADCSOC3CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC3 Trigger Source Select. Along with the SOC3 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC3 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCA 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCA 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCA 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCA 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCA 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCA 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCA 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCA 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved



**Table 13-32. ADCSOC3CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC3 Channel Select. Selects the channel to be converted when SOC3 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC3 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.21 ADCSOC4CTL Register (Offset = 18h) [reset = 0h]**

ADCSOC4CTL is shown in [Figure 13-41](#) and described in [Table 13-33](#).

Return to the [Summary Table](#).

ADC SOC4 Control Register

**Figure 13-41. ADCSOC4CTL Register**

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

**Table 13-33. ADCSOC4CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC4 Trigger Source Select. Along with the SOC4 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC4 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-33. ADCSOC4CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC4 Channel Select. Selects the channel to be converted when SOC4 is received by the ADC.</p> <p>0h ADCIN0            1h ADCIN1            2h ADCIN2            3h ADCIN3            4h ADCIN4            5h ADCIN5            6h ADCIN6            7h ADCIN7            8h ADCIN8            9h ADCIN9            Ah ADCIN10            Bh ADCIN11            Ch ADCIN12            Dh ADCIN13            Eh ADCIN14            Fh ADCIN15            Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC4 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.</p> <p>000h Sample window is 1 system clock cycle wide            001h Sample window is 2 system clock cycles wide            002h Sample window is 3 system clock cycles wide            ...            1FFh Sample window is 512 system clock cycles wide            Reset type: SYSRSn</p>

**13.16.2.22 ADCSOC5CTL Register (Offset = 1Ah) [reset = 0h]**

ADCSOC5CTL is shown in [Figure 13-42](#) and described in [Table 13-34](#).

Return to the [Summary Table](#).

ADC SOC5 Control Register

**Figure 13-42. ADCSOC5CTL Register**

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

**Table 13-34. ADCSOC5CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC5 Trigger Source Select. Along with the SOC5 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC5 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-34. ADCSOC5CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC5 Channel Select. Selects the channel to be converted when SOC5 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC5 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.23 ADCSOC6CTL Register (Offset = 1Ch) [reset = 0h]**

ADCSOC6CTL is shown in [Figure 13-43](#) and described in [Table 13-35](#).

Return to the [Summary Table](#).

ADC SOC6 Control Register

**Figure 13-43. ADCSOC6CTL Register**

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

**Table 13-35. ADCSOC6CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC6 Trigger Source Select. Along with the SOC6 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC6 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-35. ADCSOC6CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC6 Channel Select. Selects the channel to be converted when SOC6 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC6 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.24 ADCSOC7CTL Register (Offset = 1Eh) [reset = 0h]**

ADCSOC7CTL is shown in [Figure 13-44](#) and described in [Table 13-36](#).

Return to the [Summary Table](#).

ADC SOC7 Control Register

**Figure 13-44. ADCSOC7CTL Register**

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

**Table 13-36. ADCSOC7CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC7 Trigger Source Select. Along with the SOC7 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC7 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCA 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCA 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCA 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCA 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCA 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCA 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCA 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCA 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved



**Table 13-36. ADCSOC7CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC7 Channel Select. Selects the channel to be converted when SOC7 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC7 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.25 ADCSOC8CTL Register (Offset = 20h) [reset = 0h]**

 ADCSOC8CTL is shown in [Figure 13-45](#) and described in [Table 13-37](#).

 Return to the [Summary Table](#).

ADC SOC8 Control Register

**Figure 13-45. ADCSOC8CTL Register**

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

**Table 13-37. ADCSOC8CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC8 Trigger Source Select. Along with the SOC8 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC8 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-37. ADCSOC8CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC8 Channel Select. Selects the channel to be converted when SOC8 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC8 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.26 ADCSOC9CTL Register (Offset = 22h) [reset = 0h]**

 ADCSOC9CTL is shown in [Figure 13-46](#) and described in [Table 13-38](#).

 Return to the [Summary Table](#).

ADC SOC9 Control Register

**Figure 13-46. ADCSOC9CTL Register**

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

**Table 13-38. ADCSOC9CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC9 Trigger Source Select. Along with the SOC9 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC9 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-38. ADCSOC9CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC9 Channel Select. Selects the channel to be converted when SOC9 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC9 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.27 ADCSOC10CTL Register (Offset = 24h) [reset = 0h]**

 ADCSOC10CTL is shown in [Figure 13-47](#) and described in [Table 13-39](#).

 Return to the [Summary Table](#).

ADC SOC10 Control Register

**Figure 13-47. ADCSOC10CTL Register**

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

**Table 13-39. ADCSOC10CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC10 Trigger Source Select. Along with the SOC10 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC10 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCA 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCA 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCA 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCA 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCA 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCA 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCA 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCA 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-39. ADCSOC10CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC10 Channel Select. Selects the channel to be converted when SOC10 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC10 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.28 ADCSOC11CTL Register (Offset = 26h) [reset = 0h]**

 ADCSOC11CTL is shown in [Figure 13-48](#) and described in [Table 13-40](#).

 Return to the [Summary Table](#).

ADC SOC11 Control Register

**Figure 13-48. ADCSOC11CTL Register**

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

**Table 13-40. ADCSOC11CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC11 Trigger Source Select. Along with the SOC11 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC11 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved



**Table 13-40. ADCSOC11CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC11 Channel Select. Selects the channel to be converted when SOC11 is received by the ADC.</p> <p>0h ADCIN0            1h ADCIN1            2h ADCIN2            3h ADCIN3            4h ADCIN4            5h ADCIN5            6h ADCIN6            7h ADCIN7            8h ADCIN8            9h ADCIN9            Ah ADCIN10            Bh ADCIN11            Ch ADCIN12            Dh ADCIN13            Eh ADCIN14            Fh ADCIN15            Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC11 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.</p> <p>000h Sample window is 1 system clock cycle wide            001h Sample window is 2 system clock cycles wide            002h Sample window is 3 system clock cycles wide            ...            1FFh Sample window is 512 system clock cycles wide            Reset type: SYSRSn</p>

**13.16.2.29 ADCSOC12CTL Register (Offset = 28h) [reset = 0h]**

 ADCSOC12CTL is shown in [Figure 13-49](#) and described in [Table 13-41](#).

 Return to the [Summary Table](#).

ADC SOC12 Control Register

**Figure 13-49. ADCSOC12CTL Register**

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

**Table 13-41. ADCSOC12CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC12 Trigger Source Select. Along with the SOC12 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC12 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-41. ADCSOC12CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC12 Channel Select. Selects the channel to be converted when SOC12 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC12 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.30 ADCSOC13CTL Register (Offset = 2Ah) [reset = 0h]**

 ADCSOC13CTL is shown in [Figure 13-50](#) and described in [Table 13-42](#).

 Return to the [Summary Table](#).

ADC SOC13 Control Register

**Figure 13-50. ADCSOC13CTL Register**

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

**Table 13-42. ADCSOC13CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC13 Trigger Source Select. Along with the SOC13 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC13 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCA 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCA 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCA 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCA 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCA 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCA 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCA 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCA 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-42. ADCSOC13CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC13 Channel Select. Selects the channel to be converted when SOC13 is received by the ADC.</p> <p>0h ADCIN0  1h ADCIN1  2h ADCIN2  3h ADCIN3  4h ADCIN4  5h ADCIN5  6h ADCIN6  7h ADCIN7  8h ADCIN8  9h ADCIN9  Ah ADCIN10  Bh ADCIN11  Ch ADCIN12  Dh ADCIN13  Eh ADCIN14  Fh ADCIN15  Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC13 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.</p> <p>000h Sample window is 1 system clock cycle wide  001h Sample window is 2 system clock cycles wide  002h Sample window is 3 system clock cycles wide  ...  1FFh Sample window is 512 system clock cycles wide  Reset type: SYSRSn</p>

**13.16.2.31 ADCSOC14CTL Register (Offset = 2Ch) [reset = 0h]**

 ADCSOC14CTL is shown in [Figure 13-51](#) and described in [Table 13-43](#).

 Return to the [Summary Table](#).

ADC SOC14 Control Register

**Figure 13-51. ADCSOC14CTL Register**

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

**Table 13-43. ADCSOC14CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC14 Trigger Source Select. Along with the SOC14 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC14 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved

**Table 13-43. ADCSOC14CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC14 Channel Select. Selects the channel to be converted when SOC14 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC14 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

**13.16.2.32 ADCSOC15CTL Register (Offset = 2Eh) [reset = 0h]**

 ADCSOC15CTL is shown in [Figure 13-52](#) and described in [Table 13-44](#).

 Return to the [Summary Table](#).

ADC SOC15 Control Register

**Figure 13-52. ADCSOC15CTL Register**

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

**Table 13-44. ADCSOC15CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	TRIGSEL	R/W	0h	SOC15 Trigger Source Select. Along with the SOC15 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC15 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. 00h ADCTRIG0 - Software only 01h ADCTRIG1 - CPU1 Timer 0, TINT0n 02h ADCTRIG2 - CPU1 Timer 1, TINT1n 03h ADCTRIG3 - CPU1 Timer 2, TINT2n 04h ADCTRIG4 - GPIO, ADCEXTSOC 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h ADCTRIG19 - ePWM8, ADCSOCA 14h ADCTRIG20 - ePWM8, ADCSOCB 15h - 1Fh - Reserved Reset type: SYSRSn
19	RESERVED	R	0h	Reserved



**Table 13-44. ADCSOC15CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	SOC15 Channel Select. Selects the channel to be converted when SOC15 is received by the ADC.  0h ADCIN0 1h ADCIN1 2h ADCIN2 3h ADCIN3 4h ADCIN4 5h ADCIN5 6h ADCIN6 7h ADCIN7 8h ADCIN8 9h ADCIN9 Ah ADCIN10 Bh ADCIN11 Ch ADCIN12 Dh ADCIN13 Eh ADCIN14 Fh ADCIN15 Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	SOC15 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration.  000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 13.16.2.33 ADCEVTSTAT Register (Offset = 30h) [reset = 0h]

ADCEVTSTAT is shown in [Figure 13-53](#) and described in [Table 13-45](#).

Return to the [Summary Table](#).

ADC Event Status Register

**Figure 13-53. ADCEVTSTAT Register**

15		14		13		12		11		10		9		8	
RESERVED	PPB4ZERO	PPB4TRIPLO	PPB4TRIPHI	RESERVED	PPB3ZERO	PPB3TRIPLO	PPB3TRIPHI	RESERVED	PPB2ZERO	PPB2TRIPLO	PPB2TRIPHI	RESERVED	PPB1ZERO	PPB1TRIPLO	PPB1TRIPHI
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7		6		5		4		3		2		1		0	
RESERVED	PPB2ZERO	PPB2TRIPLO	PPB2TRIPHI	RESERVED	PPB1ZERO	PPB1TRIPLO	PPB1TRIPHI	RESERVED	PPB0ZERO	PPB0TRIPLO	PPB0TRIPHI	RESERVED	PPB0ZERO	PPB0TRIPLO	PPB0TRIPHI
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 13-45. ADCEVTSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	PPB4ZERO	R	0h	Post Processing Block 4 Zero Crossing Flag. When set indicates the ADCPPB4RESULT register has changed sign. This bit is gated by EOC signal. Reset type: SYSRSn
13	PPB4TRIPLO	R	0h	Post Processing Block 4 Trip Low Flag. When set indicates a digital compare trip low event has occurred. Reset type: SYSRSn
12	PPB4TRIPHI	R	0h	Post Processing Block 4 Trip High Flag. When set indicates a digital compare trip high event has occurred. Reset type: SYSRSn
11	RESERVED	R	0h	Reserved
10	PPB3ZERO	R	0h	Post Processing Block 3 Zero Crossing Flag. When set indicates the ADCPPB3RESULT register has changed sign. This bit is gated by EOC signal. Reset type: SYSRSn
9	PPB3TRIPLO	R	0h	Post Processing Block 3 Trip Low Flag. When set indicates a digital compare trip low event has occurred. Reset type: SYSRSn
8	PPB3TRIPHI	R	0h	Post Processing Block 3 Trip High Flag. When set indicates a digital compare trip high event has occurred. Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6	PPB2ZERO	R	0h	Post Processing Block 2 Zero Crossing Flag. When set indicates the ADCPPB2RESULT register has changed sign. This bit is gated by EOC signal. Reset type: SYSRSn
5	PPB2TRIPLO	R	0h	Post Processing Block 2 Trip Low Flag. When set indicates a digital compare trip low event has occurred. Reset type: SYSRSn
4	PPB2TRIPHI	R	0h	Post Processing Block 2 Trip High Flag. When set indicates a digital compare trip high event has occurred. Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	PPB1ZERO	R	0h	Post Processing Block 1 Zero Crossing Flag. When set indicates the ADCPPB1RESULT register has changed sign. This bit is gated by EOC signal. Reset type: SYSRSn

**Table 13-45. ADCEVTSTAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	PPB1TRIPLO	R	0h	Post Processing Block 1 Trip Low Flag. When set indicates a digital compare trip low event has occurred. Reset type: SYSRSn
0	PPB1TRIPHI	R	0h	Post Processing Block 1 Trip High Flag. When set indicates a digital compare trip high event has occurred. Reset type: SYSRSn

**13.16.2.34 ADCEVTCLR Register (Offset = 32h) [reset = 0h]**

ADCEVTCLR is shown in [Figure 13-54](#) and described in [Table 13-46](#).

Return to the [Summary Table](#).

ADC Event Clear Register

**Figure 13-54. ADCEVTCLR Register**

15		14		13		12		11		10		9		8	
RESERVED	PPB4ZERO	PPB4TRIPLO	PPB4TRIPHI	RESERVED	PPB3ZERO	PPB3TRIPLO	PPB3TRIPHI	RESERVED	PPB2ZERO	PPB2TRIPLO	PPB2TRIPHI	RESERVED	PPB1ZERO	PPB1TRIPLO	PPB1TRIPHI
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h
7		6		5		4		3		2		1		0	
RESERVED	PPB2ZERO	PPB2TRIPLO	PPB2TRIPHI	RESERVED	PPB1ZERO	PPB1TRIPLO	PPB1TRIPHI	RESERVED	PPB0ZERO	PPB0TRIPLO	PPB0TRIPHI	RESERVED	PPB0ZERO	PPB0TRIPLO	PPB0TRIPHI
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-46. ADCEVTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	PPB4ZERO	R/W	0h	Post Processing Block 4 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register. Reset type: SYSRSn
13	PPB4TRIPLO	R/W	0h	Post Processing Block 4 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register. Reset type: SYSRSn
12	PPB4TRIPHI	R/W	0h	Post Processing Block 4 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register. Reset type: SYSRSn
11	RESERVED	R	0h	Reserved
10	PPB3ZERO	R/W	0h	Post Processing Block 3 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register. Reset type: SYSRSn
9	PPB3TRIPLO	R/W	0h	Post Processing Block 3 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register. Reset type: SYSRSn
8	PPB3TRIPHI	R/W	0h	Post Processing Block 3 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register. Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6	PPB2ZERO	R/W	0h	Post Processing Block 2 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register. Reset type: SYSRSn
5	PPB2TRIPLO	R/W	0h	Post Processing Block 2 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register. Reset type: SYSRSn
4	PPB2TRIPHI	R/W	0h	Post Processing Block 2 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register. Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	PPB1ZERO	R/W	0h	Post Processing Block 1 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register. Reset type: SYSRSn
1	PPB1TRIPLO	R/W	0h	Post Processing Block 1 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register. Reset type: SYSRSn

**Table 13-46. ADCEVTCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	PPB1TRIPHI	R/W	0h	Post Processing Block 1 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register. Reset type: SYSRSn

**13.16.2.35 ADCEVTSEL Register (Offset = 34h) [reset = 0h]**

 ADCEVTSEL is shown in [Figure 13-55](#) and described in [Table 13-47](#).

 Return to the [Summary Table](#).

ADC Event Selection Register

**Figure 13-55. ADCEVTSEL Register**

15	14	13	12	11	10	9	8
RESERVED	PPB4ZERO	PPB4TRIPLO	PPB4TRIPHI	RESERVED	PPB3ZERO	PPB3TRIPLO	PPB3TRIPHI
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED	PPB2ZERO	PPB2TRIPLO	PPB2TRIPHI	RESERVED	PPB1ZERO	PPB1TRIPLO	PPB1TRIPHI
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-47. ADCEVTSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	PPB4ZERO	R/W	0h	Post Processing Block 4 Zero Crossing Event Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
13	PPB4TRIPLO	R/W	0h	Post Processing Block 4 Trip Low Event Enable. Setting this bit allows the corresponding rising trip low flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
12	PPB4TRIPHI	R/W	0h	Post Processing Block 4 Trip High Event Enable. Setting this bit allows the corresponding rising trip high flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
11	RESERVED	R	0h	Reserved
10	PPB3ZERO	R/W	0h	Post Processing Block 3 Zero Crossing Event Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
9	PPB3TRIPLO	R/W	0h	Post Processing Block 3 Trip Low Event Enable. Setting this bit allows the corresponding rising trip low flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
8	PPB3TRIPHI	R/W	0h	Post Processing Block 3 Trip High Event Enable. Setting this bit allows the corresponding rising trip high flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6	PPB2ZERO	R/W	0h	Post Processing Block 2 Zero Crossing Event Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn

**Table 13-47. ADCEVTSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	PPB2TRIPLO	R/W	0h	Post Processing Block 2 Trip Low Event Enable. Setting this bit allows the corresponding rising trip low flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
4	PPB2TRIPHI	R/W	0h	Post Processing Block 2 Trip High Event Enable. Setting this bit allows the corresponding rising trip high flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	PPB1ZERO	R/W	0h	Post Processing Block 1 Zero Crossing Event Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
1	PPB1TRIPLO	R/W	0h	Post Processing Block 1 Trip Low Event Enable. Setting this bit allows the corresponding rising trip low flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
0	PPB1TRIPHI	R/W	0h	Post Processing Block 1 Trip High Event Enable. Setting this bit allows the corresponding rising trip high flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn

**13.16.2.36 ADCEVTINTSEL Register (Offset = 36h) [reset = 0h]**

 ADCEVTINTSEL is shown in [Figure 13-56](#) and described in [Table 13-48](#).

 Return to the [Summary Table](#).

ADC Event Interrupt Selection Register

**Figure 13-56. ADCEVTINTSEL Register**

15		14		13		12		11		10		9		8	
RESERVED	PPB4ZERO	PPB4TRIPLO	PPB4TRIPHI	RESERVED	PPB3ZERO	PPB3TRIPLO	PPB3TRIPHI	RESERVED	PPB2ZERO	PPB2TRIPLO	PPB2TRIPHI	RESERVED	PPB1ZERO	PPB1TRIPLO	PPB1TRIPHI
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h
7		6		5		4		3		2		1		0	
RESERVED	PPB2ZERO	PPB2TRIPLO	PPB2TRIPHI	RESERVED	PPB1ZERO	PPB1TRIPLO	PPB1TRIPHI	RESERVED	PPB0ZERO	PPB0TRIPLO	PPB0TRIPHI	RESERVED	PPB0ZERO	PPB0TRIPLO	PPB0TRIPHI
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-48. ADCEVTINTSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	PPB4ZERO	R/W	0h	Post Processing Block 4 Zero Crossing Interrupt Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
13	PPB4TRIPLO	R/W	0h	Post Processing Block 4 Trip Low Interrupt Enable. Setting this bit allows the corresponding rising trip low flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
12	PPB4TRIPHI	R/W	0h	Post Processing Block 4 Trip High Interrupt Enable. Setting this bit allows the corresponding rising trip high flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
11	RESERVED	R	0h	Reserved
10	PPB3ZERO	R/W	0h	Post Processing Block 3 Zero Crossing Interrupt Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
9	PPB3TRIPLO	R/W	0h	Post Processing Block 3 Trip Low Interrupt Enable. Setting this bit allows the corresponding rising trip low flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
8	PPB3TRIPHI	R/W	0h	Post Processing Block 3 Trip High Interrupt Enable. Setting this bit allows the corresponding rising trip high flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6	PPB2ZERO	R/W	0h	Post Processing Block 2 Zero Crossing Interrupt Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn



**Table 13-48. ADCEVTINTSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	PPB2TRIPLO	R/W	0h	Post Processing Block 2 Trip Low Interrupt Enable. Setting this bit allows the corresponding rising trip low flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
4	PPB2TRIPHI	R/W	0h	Post Processing Block 2 Trip High Interrupt Enable. Setting this bit allows the corresponding rising trip high flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	PPB1ZERO	R/W	0h	Post Processing Block 1 Zero Crossing Interrupt Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
1	PPB1TRIPLO	R/W	0h	Post Processing Block 1 Trip Low Interrupt Enable. Setting this bit allows the corresponding rising trip low flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
0	PPB1TRIPHI	R/W	0h	Post Processing Block 1 Trip High Interrupt Enable. Setting this bit allows the corresponding rising trip high flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn

**13.16.2.37 ADCOSDETECT Register (Offset = 38h) [reset = 0h]**

ADCOSDETECT is shown in [Figure 13-57](#) and described in [Table 13-49](#).

Return to the [Summary Table](#).

ADC Open and Shorts Detect Register

**Figure 13-57. ADCOSDETECT Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				DETECTCFG			
R-0h				R/W-0h			

**Table 13-49. ADCOSDETECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2-0	DETECTCFG	R/W	0h	<p>ADC Opens and Shorts Detect Configuration. This bit field defines the open/shorts detection circuit state.</p> <p>0h Open/Shorts detection circuit is disabled.</p> <p>1h Open/Shorts detection circuit is enabled at zero scale.</p> <p>2h Open/Shorts detection circuit is enabled at full scale.</p> <p>3h Open/Shorts detection circuit is enabled at (nominal) 5/12 scale.</p> <p>4h Open/Shorts detection circuit is enabled at (nominal) 7/12 scale.</p> <p>5h Open/Shorts detection circuit is enabled with a (nominal) 5K pulldown to VSSA.</p> <p>6h Open/Shorts detection circuit is enabled with a (nominal) 5K pullup to VDDA.</p> <p>7h Open/Shorts detection circuit is enabled with a (nominal) 7K pulldown to VSSA.</p> <p>Reset type: SYSRSn</p>

**13.16.2.38 ADCCOUNTER Register (Offset = 39h) [reset = 0h]**

ADCCOUNTER is shown in [Figure 13-58](#) and described in [Table 13-50](#).

Return to the [Summary Table](#).

ADC Counter Register

**Figure 13-58. ADCCOUNTER Register**

15	14	13	12	11	10	9	8
RESERVED				FREECOUNT			
R-0h				R-0h			
7	6	5	4	3	2	1	0
FREECOUNT							
R-0h							

**Table 13-50. ADCCOUNTER Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	FREECOUNT	R	0h	ADC Free Running Counter Value. This bit field reflects the status of the free running ADC counter. Reset type: SYSRSn

### 13.16.2.39 ADCREV Register (Offset = 3Ah) [reset = 5h]

ADCREV is shown in [Figure 13-59](#) and described in [Table 13-51](#).

Return to the [Summary Table](#).

ADC Revision Register

**Figure 13-59. ADCREV Register**

15	14	13	12	11	10	9	8
REV							
R-0h							
7	6	5	4	3	2	1	0
TYPE							
R-5h							

**Table 13-51. ADCREV Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	REV	R	0h	ADC Revision. To allow documentation of differences between revisions. First version is labeled as 00h. Reset type: SYSRSn
7-0	TYPE	R	5h	ADC Type. Always set to 5 for this ADC. Reset type: SYSRSn

**13.16.2.40 ADCOFFTRIM Register (Offset = 3Bh) [reset = 0h]**

ADCOFFTRIM is shown in [Figure 13-60](#) and described in [Table 13-52](#).

Return to the [Summary Table](#).

ADC Offset Trim Register

**Figure 13-60. ADCOFFTRIM Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
OFFTRIM							
R/W-0h							

**Table 13-52. ADCOFFTRIM Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	OFFTRIM	R/W	0h	<p>ADC Offset Trim</p> <p>Adjusts the conversion results of the converter up or down to account for offset error in the ADC. A factory trim setting will be loaded during device boot.</p> <p>Offset can be corrected in the range of +7 to -8 LSBs. Value is <math>16 \times \text{Offset}</math> in 8-bit 2's complement:</p> <p>7 LSB (<math>16 \times 7</math>) = 112            6 LSB (<math>16 \times 6</math>) = 96            5 LSB (<math>16 \times 5</math>) = 80            4 LSB (<math>16 \times 4</math>) = 64            3 LSB (<math>16 \times 3</math>) = 48            2 LSB (<math>16 \times 2</math>) = 32            1 LSB (<math>16 \times 1</math>) = 16            0 LSB (<math>16 \times 0</math>) = 0            -1 LSB (<math>16 \times (-1)</math>) = 240            :            :            -7LSB(<math>16 \times (-7)</math>) = 144</p> <p>Reset type: SYSRSn</p>

**13.16.2.41 ADCPPB1CONFIG Register (Offset = 40h) [reset = 0h]**

 ADCPPB1CONFIG is shown in [Figure 13-61](#) and described in [Table 13-53](#).

 Return to the [Summary Table](#).

ADC PPB1 Config Register

**Figure 13-61. ADCPPB1CONFIG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		CBCEN	TWOSCOMPE N	CONFIG			
R-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 13-53. ADCPPB1CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	Reserved
5	CBCEN	R/W	0h	ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present. Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4	TWOSCOMPEN	R/W	0h	ADC Post Processing Block 1 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB1RESULT register.  0 ADCPPB1RESULT = ADCRESULTx - ADCPPB1OFFREF 1 ADCPPB1RESULT = ADCPPB1OFFREF - ADCRESULTx Reset type: SYSRSn

**Table 13-53. ADCPB1CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	<p>ADC Post Processing Block 1 Configuration. This bit field defines which SOC/EOC/RESULT is associated with this post processing block.</p> <p>0000 SOC0/EOC0/RESULT0 is associated with post processing block 1</p> <p>0001 SOC1/EOC1/RESULT1 is associated with post processing block 1</p> <p>0010 SOC2/EOC2/RESULT2 is associated with post processing block 1</p> <p>0011 SOC3/EOC3/RESULT3 is associated with post processing block 1</p> <p>0100 SOC4/EOC4/RESULT4 is associated with post processing block 1</p> <p>0101 SOC5/EOC5/RESULT5 is associated with post processing block 1</p> <p>0110 SOC6/EOC6/RESULT6 is associated with post processing block 1</p> <p>0111 SOC7/EOC7/RESULT7 is associated with post processing block 1</p> <p>1000 SOC8/EOC8/RESULT8 is associated with post processing block 1</p> <p>1001 SOC9/EOC9/RESULT9 is associated with post processing block 1</p> <p>1010 SOC10/EOC10/RESULT10 is associated with post processing block 1</p> <p>1011 SOC11/EOC11/RESULT11 is associated with post processing block 1</p> <p>1100 SOC12/EOC12/RESULT12 is associated with post processing block 1</p> <p>1101 SOC13/EOC13/RESULT13 is associated with post processing block 1</p> <p>1110 SOC14/EOC14/RESULT14 is associated with post processing block 1</p> <p>1111 SOC15/EOC15/RESULT15 is associated with post processing block 1</p> <p>Reset type: SYSRSn</p>

**13.16.2.42 ADCPPB1STAMP Register (Offset = 41h) [reset = 0h]**

ADCPPB1STAMP is shown in [Figure 13-62](#) and described in [Table 13-54](#).

Return to the [Summary Table](#).

ADC PPB1 Sample Delay Time Stamp Register

**Figure 13-62. ADCPPB1STAMP Register**

15	14	13	12	11	10	9	8
RESERVED				DLYSTAMP			
R-0h				R-0h			
7	6	5	4	3	2	1	0
DLYSTAMP							
R-0h							

**Table 13-54. ADCPPB1STAMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 1 Delay Time Stamp. When an SOC starts sampling the value contained in REQSTAMP is subtracted from the value in ADCCOUNTER.FREECOUNT and loaded into this bit field, thereby giving the number of system clock cycles delay between the SOC trigger and the actual start of the sample. Reset type: SYSRSn



**13.16.2.43 ADCPPB1OFFCAL Register (Offset = 42h) [reset = 0h]**

ADCPPB1OFFCAL is shown in [Figure 13-63](#) and described in [Table 13-55](#).

Return to the [Summary Table](#).

ADC PPB1 Offset Calibration Register

**Figure 13-63. ADCPPB1OFFCAL Register**

15	14	13	12	11	10	9	8
RESERVED						OFFCAL	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
OFFCAL							
R/W-0h							

**Table 13-55. ADCPPB1OFFCAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OFFCAL	R/W	0h	<p>ADC Post Processing Block 1 Offset Correction. This bit field can be used to digitally remove any system level offset inherent in the ADCIN circuit. This 10-bit signed value is subtracted from the ADC output before being stored in the ADCRESULT register.</p> <p>000h No change. The ADC output is stored directly into ADCRESULT.</p> <p>001h ADC output - 1 is stored into ADCRESULT.</p> <p>002h ADC output - 2 is stored into ADCRESULT.</p> <p>...</p> <p>200h ADC output + 512 is stored into ADCRESULT.</p> <p>...</p> <p>3FFh ADC output + 1 is stored into ADCRESULT.</p> <p>NOTE: In 16-bit mode, the subtraction will saturate at 0000h and FFFFh before being stored into the ADCRESULT register. In 12-bit mode, the subtraction will saturate at 0000h and 0FFFh before being stored into the ADCRESULT register.</p> <p>Reset type: SYSRSn</p>

**13.16.2.44 ADCPPB1OFFREF Register (Offset = 43h) [reset = 0h]**

ADCPPB1OFFREF is shown in [Figure 13-64](#) and described in [Table 13-56](#).

Return to the [Summary Table](#).

ADC PPB1 Offset Reference Register

**Figure 13-64. ADCPPB1OFFREF Register**

15	14	13	12	11	10	9	8
OFFREF							
R/W-0h							
7	6	5	4	3	2	1	0
OFFREF							
R/W-0h							

**Table 13-56. ADCPPB1OFFREF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 1 Offset Correction. This bit field can be used to either calculate the feedback error or convert a unipolar signal to bipolar by subtracting a reference value. This 16-bit unsigned value is subtracted from the ADCRESULT register before being passed through an optional two's complement function and stored in the ADCPPB1RESULT register. This subtraction is not saturated.</p> <p>0000h No change. The ADCRESULT value is passed on.                      0001h ADCRESULT - 1 is passed on.                      0002h ADCRESULT - 2 is passed on.                      ...                      8000h ADCRESULT - 32,768 is passed on.                      ...                      FFFFh ADCRESULT - 65,535 is passed on.</p> <p>NOTE: In 12-bit mode the size of this register does not change from 16-bits. It is the user's responsibility to ensure that only a 12-bit value is written to this register when in 12-bit mode.                      Reset type: SYSRSn</p>

**13.16.2.45 ADCPPB1TRIPHI Register (Offset = 44h) [reset = 0h]**

ADCPPB1TRIPHI is shown in [Figure 13-65](#) and described in [Table 13-57](#).

Return to the [Summary Table](#).

ADC PPB1 Trip High Register

**Figure 13-65. ADCPPB1TRIPHI Register**

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

**Table 13-57. ADCPPB1TRIPHI Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	HSIGN	R/W	0h	High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITHI	R/W	0h	ADC Post Processing Block 1 Trip High Limit. This value sets the digital comparator trip high limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB1RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB1RESULT register. Reset type: SYSRSn

**13.16.2.46 ADCPPB1TRIPLO Register (Offset = 46h) [reset = 0h]**

ADCPPB1TRIPLO is shown in [Figure 13-66](#) and described in [Table 13-58](#).

Return to the [Summary Table](#).

ADC PPB1 Trip Low/Trigger Time Stamp Register

**Figure 13-66. ADCPPB1TRIPLO Register**

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

**Table 13-58. ADCPPB1TRIPLO Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 1 Request Time Stamp. When a trigger sets the associated SOC flag in the ADCSOCFLG1 register the value of ADCCOUNTER.FREECOUNT is loaded into this bit field. Reset type: SYSRSn
19-17	RESERVED	R	0h	Reserved
16	LSIGN	R/W	0h	Low Limit Sign Bit. This is the sign bit (17th bit) to the LIMITLO bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITLO	R/W	0h	ADC Post Processing Block 1 Trip Low Limit. This value sets the digital comparator trip low limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB1RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB1RESULT register. Reset type: SYSRSn

### 13.16.2.47 ADCPPB2CONFIG Register (Offset = 48h) [reset = 0h]

ADCPPB2CONFIG is shown in [Figure 13-67](#) and described in [Table 13-59](#).

Return to the [Summary Table](#).

ADC PPB2 Config Register

**Figure 13-67. ADCPPB2CONFIG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		CBCEN	TWOSCOMPE N	CONFIG			
R-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 13-59. ADCPPB2CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	Reserved
5	CBCEN	R/W	0h	ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present. Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4	TWOSCOMPEN	R/W	0h	ADC Post Processing Block 2 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB2RESULT register.  0 ADCPPB2RESULT = ADCRESULTx - ADCPPB2OFFREF 1 ADCPPB2RESULT = ADCPPB2OFFREF - ADCRESULTx Reset type: SYSRSn

**Table 13-59. ADCPB2CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	ADC Post Processing Block 2 Configuration. This bit field defines which SOC/EOC/RESULT is associated with this post processing block.  0000 SOC0/EOC0/RESULT0 is associated with post processing block 2 0001 SOC1/EOC1/RESULT1 is associated with post processing block 2 0010 SOC2/EOC2/RESULT2 is associated with post processing block 2 0011 SOC3/EOC3/RESULT3 is associated with post processing block 2 0100 SOC4/EOC4/RESULT4 is associated with post processing block 2 0101 SOC5/EOC5/RESULT5 is associated with post processing block 2 0110 SOC6/EOC6/RESULT6 is associated with post processing block 2 0111 SOC7/EOC7/RESULT7 is associated with post processing block 2 1000 SOC8/EOC8/RESULT8 is associated with post processing block 2 1001 SOC9/EOC9/RESULT9 is associated with post processing block 2 1010 SOC10/EOC10/RESULT10 is associated with post processing block 2 1011 SOC11/EOC11/RESULT11 is associated with post processing block 2 1100 SOC12/EOC12/RESULT12 is associated with post processing block 2 1101 SOC13/EOC13/RESULT13 is associated with post processing block 2 1110 SOC14/EOC14/RESULT14 is associated with post processing block 2 1111 SOC15/EOC15/RESULT15 is associated with post processing block 2  Reset type: SYSRSn

**13.16.2.48 ADCPPB2STAMP Register (Offset = 49h) [reset = 0h]**

ADCPPB2STAMP is shown in [Figure 13-68](#) and described in [Table 13-60](#).

Return to the [Summary Table](#).

ADC PPB2 Sample Delay Time Stamp Register

**Figure 13-68. ADCPPB2STAMP Register**

15	14	13	12	11	10	9	8
RESERVED				DLYSTAMP			
R-0h				R-0h			
7	6	5	4	3	2	1	0
DLYSTAMP							
R-0h							

**Table 13-60. ADCPPB2STAMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 2 Delay Time Stamp. When an SOC starts sampling the value contained in REQSTAMP is subtracted from the value in ADCCOUNTER.FREECOUNT and loaded into this bit field, thereby giving the number of system clock cycles delay between the SOC trigger and the actual start of the sample. Reset type: SYSRSn

**13.16.2.49 ADCPPB2OFFCAL Register (Offset = 4Ah) [reset = 0h]**

ADCPPB2OFFCAL is shown in [Figure 13-69](#) and described in [Table 13-61](#).

Return to the [Summary Table](#).

ADC PPB2 Offset Calibration Register

**Figure 13-69. ADCPPB2OFFCAL Register**

15	14	13	12	11	10	9	8
RESERVED						OFFCAL	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
OFFCAL							
R/W-0h							

**Table 13-61. ADCPPB2OFFCAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OFFCAL	R/W	0h	<p>ADC Post Processing Block 2 Offset Correction. This bit field can be used to digitally remove any system level offset inherent in the ADCIN circuit. This 10-bit signed value is subtracted from the ADC output before being stored in the ADCRESULT register.</p> <p>000h No change. The ADC output is stored directly into ADCRESULT.</p> <p>001h ADC output - 1 is stored into ADCRESULT.</p> <p>002h ADC output - 2 is stored into ADCRESULT.</p> <p>...</p> <p>200h ADC output + 512 is stored into ADCRESULT.</p> <p>...</p> <p>3FFh ADC output + 1 is stored into ADCRESULT.</p> <p>NOTE: In 16-bit mode, the subtraction will saturate at 0000h and FFFFh before being stored into the ADCRESULT register. In 12-bit mode, the subtraction will saturate at 0000h and 0FFFh before being stored into the ADCRESULT register.</p> <p>Reset type: SYSRSn</p>



**13.16.2.50 ADCPPB2OFFREF Register (Offset = 4Bh) [reset = 0h]**

ADCPPB2OFFREF is shown in [Figure 13-70](#) and described in [Table 13-62](#).

Return to the [Summary Table](#).

ADC PPB2 Offset Reference Register

**Figure 13-70. ADCPPB2OFFREF Register**

15	14	13	12	11	10	9	8
OFFREF							
R/W-0h							
7	6	5	4	3	2	1	0
OFFREF							
R/W-0h							

**Table 13-62. ADCPPB2OFFREF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 2 Offset Correction. This bit field can be used to either calculate the feedback error or convert a unipolar signal to bipolar by subtracting a reference value. This 16-bit unsigned value is subtracted from the ADCRESULT register before being passed through an optional two's complement function and stored in the ADCPPB2RESULT register. This subtraction is not saturated.</p> <p>0000h No change. The ADCRESULT value is passed on.            0001h ADCRESULT - 1 is passed on.            0002h ADCRESULT - 2 is passed on.            ...            8000h ADCRESULT - 32,768 is passed on.            ...            FFFFh ADCRESULT - 65,535 is passed on.</p> <p>NOTE: In 12-bit mode the size of this register does not change from 16-bits. It is the user's responsibility to ensure that only a 12-bit value is written to this register when in 12-bit mode.            Reset type: SYSRSn</p>

**13.16.2.51 ADCPPB2TRIPHI Register (Offset = 4Ch) [reset = 0h]**

ADCPPB2TRIPHI is shown in [Figure 13-71](#) and described in [Table 13-63](#).

Return to the [Summary Table](#).

ADC PPB2 Trip High Register

**Figure 13-71. ADCPPB2TRIPHI Register**

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

**Table 13-63. ADCPPB2TRIPHI Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	HSIGN	R/W	0h	High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITHI	R/W	0h	ADC Post Processing Block 2 Trip High Limit. This value sets the digital comparator trip high limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB2RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB2RESULT register. Reset type: SYSRSn

**13.16.2.52 ADCPPB2TRIPLO Register (Offset = 4Eh) [reset = 0h]**

ADCPPB2TRIPLO is shown in [Figure 13-72](#) and described in [Table 13-64](#).

Return to the [Summary Table](#).

ADC PPB2 Trip Low/Trigger Time Stamp Register

**Figure 13-72. ADCPPB2TRIPLO Register**

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

**Table 13-64. ADCPPB2TRIPLO Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 2 Request Time Stamp. When a trigger sets the associated SOC flag in the ADCSOCFLG1 register the value of ADCCOUNTER.FREECOUNT is loaded into this bit field. Reset type: SYSRSn
19-17	RESERVED	R	0h	Reserved
16	LSIGN	R/W	0h	Low Limit Sign Bit. This is the sign bit (17th bit) to the LIMITLO bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITLO	R/W	0h	ADC Post Processing Block 2 Trip Low Limit. This value sets the digital comparator trip low limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB2RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB2RESULT register. Reset type: SYSRSn

**13.16.2.53 ADCPPB3CONFIG Register (Offset = 50h) [reset = 0h]**

ADCPPB3CONFIG is shown in [Figure 13-73](#) and described in [Table 13-65](#).

Return to the [Summary Table](#).

ADC PPB3 Config Register

**Figure 13-73. ADCPPB3CONFIG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		CBCEN	TWOSCOMPE N	CONFIG			
R-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 13-65. ADCPPB3CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	Reserved
5	CBCEN	R/W	0h	ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present. Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4	TWOSCOMPEN	R/W	0h	ADC Post Processing Block 3 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB3RESULT register.  0 ADCPPB3RESULT = ADCRESULTx - ADCPPB3OFFREF 1 ADCPPB3RESULT = ADCPPB3OFFREF - ADCRESULTx Reset type: SYSRSn

**Table 13-65. ADCPB3CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	<p>ADC Post Processing Block 3 Configuration. This bit field defines which SOC/EOC/RESULT is associated with this post processing block.</p> <p>0000 SOC0/EOC0/RESULT0 is associated with post processing block 3</p> <p>0001 SOC1/EOC1/RESULT1 is associated with post processing block 3</p> <p>0010 SOC2/EOC2/RESULT2 is associated with post processing block 3</p> <p>0011 SOC3/EOC3/RESULT3 is associated with post processing block 3</p> <p>0100 SOC4/EOC4/RESULT4 is associated with post processing block 3</p> <p>0101 SOC5/EOC5/RESULT5 is associated with post processing block 3</p> <p>0110 SOC6/EOC6/RESULT6 is associated with post processing block 3</p> <p>0111 SOC7/EOC7/RESULT7 is associated with post processing block 3</p> <p>1000 SOC8/EOC8/RESULT8 is associated with post processing block 3</p> <p>1001 SOC9/EOC9/RESULT9 is associated with post processing block 3</p> <p>1010 SOC10/EOC10/RESULT10 is associated with post processing block 3</p> <p>1011 SOC11/EOC11/RESULT11 is associated with post processing block 3</p> <p>1100 SOC12/EOC12/RESULT12 is associated with post processing block 3</p> <p>1101 SOC13/EOC13/RESULT13 is associated with post processing block 3</p> <p>1110 SOC14/EOC14/RESULT14 is associated with post processing block 3</p> <p>1111 SOC15/EOC15/RESULT15 is associated with post processing block 3</p> <p>Reset type: SYSRSn</p>

**13.16.2.54 ADCPPB3STAMP Register (Offset = 51h) [reset = 0h]**

ADCPPB3STAMP is shown in [Figure 13-74](#) and described in [Table 13-66](#).

Return to the [Summary Table](#).

ADC PPB3 Sample Delay Time Stamp Register

**Figure 13-74. ADCPPB3STAMP Register**

15	14	13	12	11	10	9	8
RESERVED				DLYSTAMP			
R-0h				R-0h			
7	6	5	4	3	2	1	0
DLYSTAMP							
R-0h							

**Table 13-66. ADCPPB3STAMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 3 Delay Time Stamp. When an SOC starts sampling the value contained in REQSTAMP is subtracted from the value in ADCCOUNTER.FREECOUNT and loaded into this bit field, thereby giving the number of system clock cycles delay between the SOC trigger and the actual start of the sample. Reset type: SYSRSn

**13.16.2.55 ADCPPB3OFFCAL Register (Offset = 52h) [reset = 0h]**

ADCPPB3OFFCAL is shown in [Figure 13-75](#) and described in [Table 13-67](#).

Return to the [Summary Table](#).

ADC PPB3 Offset Calibration Register

**Figure 13-75. ADCPPB3OFFCAL Register**

15	14	13	12	11	10	9	8
RESERVED						OFFCAL	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
OFFCAL							
R/W-0h							

**Table 13-67. ADCPPB3OFFCAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OFFCAL	R/W	0h	<p>ADC Post Processing Block 3 Offset Correction. This bit field can be used to digitally remove any system level offset inherent in the ADCIN circuit. This 10-bit signed value is subtracted from the ADC output before being stored in the ADCRESULT register.</p> <p>000h No change. The ADC output is stored directly into ADCRESULT.</p> <p>001h ADC output - 1 is stored into ADCRESULT.</p> <p>002h ADC output - 2 is stored into ADCRESULT.</p> <p>...</p> <p>200h ADC output + 512 is stored into ADCRESULT.</p> <p>...</p> <p>3FFh ADC output + 1 is stored into ADCRESULT.</p> <p>NOTE: In 16-bit mode, the subtraction will saturate at 0000h and FFFFh before being stored into the ADCRESULT register. In 12-bit mode, the subtraction will saturate at 0000h and 0FFFh before being stored into the ADCRESULT register.</p> <p>Reset type: SYSRSn</p>

**13.16.2.56 ADCPPB3OFFREF Register (Offset = 53h) [reset = 0h]**

ADCPPB3OFFREF is shown in [Figure 13-76](#) and described in [Table 13-68](#).

Return to the [Summary Table](#).

ADC PPB3 Offset Reference Register

**Figure 13-76. ADCPPB3OFFREF Register**

15	14	13	12	11	10	9	8
OFFREF							
R/W-0h							
7	6	5	4	3	2	1	0
OFFREF							
R/W-0h							

**Table 13-68. ADCPPB3OFFREF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 3 Offset Correction. This bit field can be used to either calculate the feedback error or convert a unipolar signal to bipolar by subtracting a reference value. This 16-bit unsigned value is subtracted from the ADCRESULT register before being passed through an optional two's complement function and stored in the ADCPPB3RESULT register. This subtraction is not saturated.</p> <p>0000h No change. The ADCRESULT value is passed on.            0001h ADCRESULT - 1 is passed on.            0002h ADCRESULT - 2 is passed on.            ...            8000h ADCRESULT - 32,768 is passed on.            ...            FFFFh ADCRESULT - 65,535 is passed on.</p> <p>NOTE: In 12-bit mode the size of this register does not change from 16-bits. It is the user's responsibility to ensure that only a 12-bit value is written to this register when in 12-bit mode.            Reset type: SYSRSn</p>



### 13.16.2.57 ADCPPB3TRIPHI Register (Offset = 54h) [reset = 0h]

ADCPPB3TRIPHI is shown in [Figure 13-77](#) and described in [Table 13-69](#).

Return to the [Summary Table](#).

ADC PPB3 Trip High Register

**Figure 13-77. ADCPPB3TRIPHI Register**

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

**Table 13-69. ADCPPB3TRIPHI Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	HSIGN	R/W	0h	High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITHI	R/W	0h	ADC Post Processing Block 3 Trip High Limit. This value sets the digital comparator trip high limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB3RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB3RESULT register. Reset type: SYSRSn

**13.16.2.58 ADCPPB3TRIPLO Register (Offset = 56h) [reset = 0h]**

 ADCPPB3TRIPLO is shown in [Figure 13-78](#) and described in [Table 13-70](#).

 Return to the [Summary Table](#).

ADC PPB3 Trip Low/Trigger Time Stamp Register

**Figure 13-78. ADCPPB3TRIPLO Register**

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

**Table 13-70. ADCPPB3TRIPLO Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 3 Request Time Stamp. When a trigger sets the associated SOC flag in the ADCSOCFLG1 register the value of ADCCOUNTER.FREECOUNT is loaded into this bit field. Reset type: SYSRSn
19-17	RESERVED	R	0h	Reserved
16	LSIGN	R/W	0h	Low Limit Sign Bit. This is the sign bit (17th bit) to the LIMITLO bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITLO	R/W	0h	ADC Post Processing Block 3 Trip Low Limit. This value sets the digital comparator trip low limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB3RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB3RESULT register. Reset type: SYSRSn

### 13.16.2.59 ADCPPB4CONFIG Register (Offset = 58h) [reset = 0h]

ADCPPB4CONFIG is shown in [Figure 13-79](#) and described in [Table 13-71](#).

Return to the [Summary Table](#).

ADC PPB4 Config Register

**Figure 13-79. ADCPPB4CONFIG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		CBCEN	TWOSCOMPE N	CONFIG			
R-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 13-71. ADCPPB4CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	Reserved
5	CBCEN	R/W	0h	ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present. Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4	TWOSCOMPEN	R/W	0h	ADC Post Processing Block 4 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB4RESULT register.  0 ADCPPB4RESULT = ADCRESULTx - ADCPPB4OFFREF 1 ADCPPB4RESULT = ADCPPB4OFFREF - ADCRESULTx Reset type: SYSRSn

**Table 13-71. ADCPB4CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	<p>ADC Post Processing Block 4 Configuration. This bit field defines which SOC/EOC/RESULT is associated with this post processing block.</p> <p>0000 SOC0/EOC0/RESULT0 is associated with post processing block 4</p> <p>0001 SOC1/EOC1/RESULT1 is associated with post processing block 4</p> <p>0010 SOC2/EOC2/RESULT2 is associated with post processing block 4</p> <p>0011 SOC3/EOC3/RESULT3 is associated with post processing block 4</p> <p>0100 SOC4/EOC4/RESULT4 is associated with post processing block 4</p> <p>0101 SOC5/EOC5/RESULT5 is associated with post processing block 4</p> <p>0110 SOC6/EOC6/RESULT6 is associated with post processing block 4</p> <p>0111 SOC7/EOC7/RESULT7 is associated with post processing block 4</p> <p>1000 SOC8/EOC8/RESULT8 is associated with post processing block 4</p> <p>1001 SOC9/EOC9/RESULT9 is associated with post processing block 4</p> <p>1010 SOC10/EOC10/RESULT10 is associated with post processing block 4</p> <p>1011 SOC11/EOC11/RESULT11 is associated with post processing block 4</p> <p>1100 SOC12/EOC12/RESULT12 is associated with post processing block 4</p> <p>1101 SOC13/EOC13/RESULT13 is associated with post processing block 4</p> <p>1110 SOC14/EOC14/RESULT14 is associated with post processing block 4</p> <p>1111 SOC15/EOC15/RESULT15 is associated with post processing block 4</p> <p>Reset type: SYSRSn</p>

**13.16.2.60 ADCPPB4STAMP Register (Offset = 59h) [reset = 0h]**

ADCPPB4STAMP is shown in [Figure 13-80](#) and described in [Table 13-72](#).

Return to the [Summary Table](#).

ADC PPB4 Sample Delay Time Stamp Register

**Figure 13-80. ADCPPB4STAMP Register**

15	14	13	12	11	10	9	8
RESERVED				DLYSTAMP			
R-0h				R-0h			
7	6	5	4	3	2	1	0
DLYSTAMP							
R-0h							

**Table 13-72. ADCPPB4STAMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 4 Delay Time Stamp. When an SOC starts sampling the value contained in REQSTAMP is subtracted from the value in ADCCOUNTER.FREECOUNT and loaded into this bit field, thereby giving the number of system clock cycles delay between the SOC trigger and the actual start of the sample. Reset type: SYSRSn

**13.16.2.61 ADCPPB4OFFCAL Register (Offset = 5Ah) [reset = 0h]**

 ADCPPB4OFFCAL is shown in [Figure 13-81](#) and described in [Table 13-73](#).

 Return to the [Summary Table](#).

ADC PPB4 Offset Calibration Register

**Figure 13-81. ADCPPB4OFFCAL Register**

15	14	13	12	11	10	9	8
RESERVED						OFFCAL	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
OFFCAL							
R/W-0h							

**Table 13-73. ADCPPB4OFFCAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OFFCAL	R/W	0h	<p>ADC Post Processing Block 4 Offset Correction. This bit field can be used to digitally remove any system level offset inherent in the ADCIN circuit. This 10-bit signed value is subtracted from the ADC output before being stored in the ADCRESULT register.</p> <p>000h No change. The ADC output is stored directly into ADCRESULT.</p> <p>001h ADC output - 1 is stored into ADCRESULT.</p> <p>002h ADC output - 2 is stored into ADCRESULT.</p> <p>...</p> <p>200h ADC output + 512 is stored into ADCRESULT.</p> <p>...</p> <p>3FFh ADC output + 1 is stored into ADCRESULT.</p> <p>NOTE: In 16-bit mode, the subtraction will saturate at 0000h and FFFFh before being stored into the ADCRESULT register. In 12-bit mode, the subtraction will saturate at 0000h and 0FFFh before being stored into the ADCRESULT register.</p> <p>Reset type: SYSRSn</p>

**13.16.2.62 ADCPPB4OFFREF Register (Offset = 5Bh) [reset = 0h]**

ADCPPB4OFFREF is shown in [Figure 13-82](#) and described in [Table 13-74](#).

Return to the [Summary Table](#).

ADC PPB4 Offset Reference Register

**Figure 13-82. ADCPPB4OFFREF Register**

15	14	13	12	11	10	9	8
OFFREF							
R/W-0h							
7	6	5	4	3	2	1	0
OFFREF							
R/W-0h							

**Table 13-74. ADCPPB4OFFREF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 4 Offset Correction. This bit field can be used to either calculate the feedback error or convert a unipolar signal to bipolar by subtracting a reference value. This 16-bit unsigned value is subtracted from the ADCRESULT register before being passed through an optional two's complement function and stored in the ADCPPB4RESULT register. This subtraction is not saturated.</p> <p>0000h No change. The ADCRESULT value is passed on.            0001h ADCRESULT - 1 is passed on.            0002h ADCRESULT - 2 is passed on.            ...            8000h ADCRESULT - 32,768 is passed on.            ...            FFFFh ADCRESULT - 65,535 is passed on.</p> <p>NOTE: In 12-bit mode the size of this register does not change from 16-bits. It is the user's responsibility to ensure that only a 12-bit value is written to this register when in 12-bit mode.            Reset type: SYSRSn</p>

**13.16.2.63 ADCPPB4TRIPHI Register (Offset = 5Ch) [reset = 0h]**

ADCPPB4TRIPHI is shown in [Figure 13-83](#) and described in [Table 13-75](#).

Return to the [Summary Table](#).

ADC PPB4 Trip High Register

**Figure 13-83. ADCPPB4TRIPHI Register**

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

**Table 13-75. ADCPPB4TRIPHI Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	HSIGN	R/W	0h	High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITHI	R/W	0h	ADC Post Processing Block 4 Trip High Limit. This value sets the digital comparator trip high limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB4RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB4RESULT register. Reset type: SYSRSn



**13.16.2.64 ADCPPB4TRIPLO Register (Offset = 5Eh) [reset = 0h]**

ADCPPB4TRIPLO is shown in [Figure 13-84](#) and described in [Table 13-76](#).

Return to the [Summary Table](#).

ADC PPB4 Trip Low/Trigger Time Stamp Register

**Figure 13-84. ADCPPB4TRIPLO Register**

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

**Table 13-76. ADCPPB4TRIPLO Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 4 Request Time Stamp. When a trigger sets the associated SOC flag in the ADCSOCFLG1 register the value of ADCCOUNTER.FREECOUNT is loaded into this bit field. Reset type: SYSRSn
19-17	RESERVED	R	0h	Reserved
16	LSIGN	R/W	0h	Low Limit Sign Bit. This is the sign bit (17th bit) to the LIMITLO bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITLO	R/W	0h	ADC Post Processing Block 4 Trip Low Limit. This value sets the digital comparator trip low limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB4RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB4RESULT register. Reset type: SYSRSn

**13.16.2.65 ADCINTCYCLE Register (Offset = 6Fh) [reset = 0h]**

ADCINTCYCLE is shown in [Figure 13-85](#) and described in [Table 13-77](#).

Return to the [Summary Table](#).

ADC Early Interrupt Generation Cycle

**Figure 13-85. ADCINTCYCLE Register**

15	14	13	12	11	10	9	8
DELAY							
R/W-0h							
7	6	5	4	3	2	1	0
DELAY							
R/W-0h							

**Table 13-77. ADCINTCYCLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DELAY	R/W	0h	ADC Early Interrupt Generation Cycle Delay: Defines the delay from the fall edge of ADCSOC in terms of system clock cycles, for the interrupt to be generated. Reset type: SYSRSn

**13.16.2.66 ADCINLTRIM1 Register (Offset = 70h) [reset = X]**

ADCINLTRIM1 is shown in [Figure 13-86](#) and described in [Table 13-78](#).

Return to the [Summary Table](#).

ADC Linearity Trim 1 Register

**Figure 13-86. ADCINLTRIM1 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
INLTRIM31TO0																															
R/W-X																															

**Table 13-78. ADCINLTRIM1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	INLTRIM31TO0	R/W	X	<p>ADC Linearity Trim Bits 31-0.</p> <p>This register should not be modified unless specifically indicated by TI Errata or other documentation. Modifying the contents of this register could cause this module to operate outside of datasheet specifications.</p> <p>Reset type: SYSRSn</p>

**13.16.2.67 ADCINLTRIM2 Register (Offset = 72h) [reset = X]**

ADCINLTRIM2 is shown in [Figure 13-87](#) and described in [Table 13-79](#).

Return to the [Summary Table](#).

ADC Linearity Trim 2 Register

**Figure 13-87. ADCINLTRIM2 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
INLTRIM63TO32																															
R/W-X																															

**Table 13-79. ADCINLTRIM2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	INLTRIM63TO32	R/W	X	ADC Linearity Trim Bits 63-32.  This register should not be modified unless specifically indicated by TI Errata or other documentation. Modifying the contents of this register could cause this module to operate outside of datasheet specifications. Reset type: SYSRSn

**13.16.2.68 ADCINLTRIM3 Register (Offset = 74h) [reset = X]**

ADCINLTRIM3 is shown in [Figure 13-88](#) and described in [Table 13-80](#).

Return to the [Summary Table](#).

ADC Linearity Trim 3 Register

**Figure 13-88. ADCINLTRIM3 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
INLTRIM95TO64																															
R/W-X																															

**Table 13-80. ADCINLTRIM3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	INLTRIM95TO64	R/W	X	<p>ADC Linearity Trim Bits 95-64.</p> <p>This register should not be modified unless specifically indicated by TI Errata or other documentation. Modifying the contents of this register could cause this module to operate outside of datasheet specifications.</p> <p>Reset type: SYSRSn</p>



### 13.16.3 ADC\_RESULT\_REGS Registers

Table 13-81 lists the ADC\_RESULT\_REGS registers. All register offset addresses not listed in Table 13-81 should be considered as reserved locations and the register contents should not be modified.

**Table 13-81. ADC\_RESULT\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	ADCRESULT0	ADC Result 0 Register		<a href="#">Go</a>
1h	ADCRESULT1	ADC Result 1 Register		<a href="#">Go</a>
2h	ADCRESULT2	ADC Result 2 Register		<a href="#">Go</a>
3h	ADCRESULT3	ADC Result 3 Register		<a href="#">Go</a>
4h	ADCRESULT4	ADC Result 4 Register		<a href="#">Go</a>
5h	ADCRESULT5	ADC Result 5 Register		<a href="#">Go</a>
6h	ADCRESULT6	ADC Result 6 Register		<a href="#">Go</a>
7h	ADCRESULT7	ADC Result 7 Register		<a href="#">Go</a>
8h	ADCRESULT8	ADC Result 8 Register		<a href="#">Go</a>
9h	ADCRESULT9	ADC Result 9 Register		<a href="#">Go</a>
Ah	ADCRESULT10	ADC Result 10 Register		<a href="#">Go</a>
Bh	ADCRESULT11	ADC Result 11 Register		<a href="#">Go</a>
Ch	ADCRESULT12	ADC Result 12 Register		<a href="#">Go</a>
Dh	ADCRESULT13	ADC Result 13 Register		<a href="#">Go</a>
Eh	ADCRESULT14	ADC Result 14 Register		<a href="#">Go</a>
Fh	ADCRESULT15	ADC Result 15 Register		<a href="#">Go</a>
10h	ADCPPB1RESULT	ADC Post Processing Block 1 Result Register		<a href="#">Go</a>
12h	ADCPPB2RESULT	ADC Post Processing Block 2 Result Register		<a href="#">Go</a>
14h	ADCPPB3RESULT	ADC Post Processing Block 3 Result Register		<a href="#">Go</a>
16h	ADCPPB4RESULT	ADC Post Processing Block 4 Result Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 13-82 shows the codes that are used for access types in this section.

**Table 13-82. ADC\_RESULT\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 13.16.3.1 ADCRESULT0 Register (Offset = 0h) [reset = 0h]

ADCRESULT0 is shown in [Figure 13-89](#) and described in [Table 13-83](#).

Return to the [Summary Table](#).

ADC Result 0 Register

**Figure 13-89. ADCRESULT0 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-83. ADCRESULT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 0  16-bit ADC result. After the ADC completes a conversion of SOC0, the digital result is placed in this bit field. Reset type: SYSRSn



**13.16.3.2 ADCRESULT1 Register (Offset = 1h) [reset = 0h]**

ADCRESULT1 is shown in [Figure 13-90](#) and described in [Table 13-84](#).

Return to the [Summary Table](#).

ADC Result 1 Register

**Figure 13-90. ADCRESULT1 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-84. ADCRESULT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 1  16-bit ADC result. After the ADC completes a conversion of SOC1, the digital result is placed in this bit field. Reset type: SYSRSn

### 13.16.3.3 ADCRESULT2 Register (Offset = 2h) [reset = 0h]

ADCRESULT2 is shown in [Figure 13-91](#) and described in [Table 13-85](#).

Return to the [Summary Table](#).

ADC Result 2 Register

**Figure 13-91. ADCRESULT2 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-85. ADCRESULT2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 2  16-bit ADC result. After the ADC completes a conversion of SOC2, the digital result is placed in this bit field. Reset type: SYSRSn

**13.16.3.4 ADCRESULT3 Register (Offset = 3h) [reset = 0h]**

ADCRESULT3 is shown in [Figure 13-92](#) and described in [Table 13-86](#).

Return to the [Summary Table](#).

ADC Result 3 Register

**Figure 13-92. ADCRESULT3 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-86. ADCRESULT3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	<p>ADC Result 3</p> <p>16-bit ADC result. After the ADC completes a conversion of SOC3, the digital result is placed in this bit field.</p> <p>Reset type: SYSRSn</p>

**13.16.3.5 ADCRESULT4 Register (Offset = 4h) [reset = 0h]**

ADCRESULT4 is shown in [Figure 13-93](#) and described in [Table 13-87](#).

Return to the [Summary Table](#).

ADC Result 4 Register

**Figure 13-93. ADCRESULT4 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-87. ADCRESULT4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 4  16-bit ADC result. After the ADC completes a conversion of SOC4, the digital result is placed in this bit field. Reset type: SYSRSn

**13.16.3.6 ADCRESULT5 Register (Offset = 5h) [reset = 0h]**

ADCRESULT5 is shown in [Figure 13-94](#) and described in [Table 13-88](#).

Return to the [Summary Table](#).

ADC Result 5 Register

**Figure 13-94. ADCRESULT5 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-88. ADCRESULT5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 5  16-bit ADC result. After the ADC completes a conversion of SOC5, the digital result is placed in this bit field. Reset type: SYSRSn

### 13.16.3.7 ADCRESULT6 Register (Offset = 6h) [reset = 0h]

ADCRESULT6 is shown in [Figure 13-95](#) and described in [Table 13-89](#).

Return to the [Summary Table](#).

ADC Result 6 Register

**Figure 13-95. ADCRESULT6 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-89. ADCRESULT6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 6  16-bit ADC result. After the ADC completes a conversion of SOC6, the digital result is placed in this bit field. Reset type: SYSRSn

**13.16.3.8 ADCRESULT7 Register (Offset = 7h) [reset = 0h]**

ADCRESULT7 is shown in [Figure 13-96](#) and described in [Table 13-90](#).

Return to the [Summary Table](#).

ADC Result 7 Register

**Figure 13-96. ADCRESULT7 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-90. ADCRESULT7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 7  16-bit ADC result. After the ADC completes a conversion of SOC7, the digital result is placed in this bit field. Reset type: SYSRSn

### 13.16.3.9 ADCRESULT8 Register (Offset = 8h) [reset = 0h]

ADCRESULT8 is shown in [Figure 13-97](#) and described in [Table 13-91](#).

Return to the [Summary Table](#).

ADC Result 8 Register

**Figure 13-97. ADCRESULT8 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-91. ADCRESULT8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 8  16-bit ADC result. After the ADC completes a conversion of SOC8, the digital result is placed in this bit field. Reset type: SYSRSn



**13.16.3.10 ADCRESULT9 Register (Offset = 9h) [reset = 0h]**

ADCRESULT9 is shown in [Figure 13-98](#) and described in [Table 13-92](#).

Return to the [Summary Table](#).

ADC Result 9 Register

**Figure 13-98. ADCRESULT9 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-92. ADCRESULT9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 9  16-bit ADC result. After the ADC completes a conversion of SOC9, the digital result is placed in this bit field. Reset type: SYSRSn

**13.16.3.11 ADCRESULT10 Register (Offset = Ah) [reset = 0h]**

ADCRESULT10 is shown in [Figure 13-99](#) and described in [Table 13-93](#).

Return to the [Summary Table](#).

ADC Result 10 Register

**Figure 13-99. ADCRESULT10 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-93. ADCRESULT10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 10  16-bit ADC result. After the ADC completes a conversion of SOC10, the digital result is placed in this bit field. Reset type: SYSRSn

**13.16.3.12 ADCRESULT11 Register (Offset = Bh) [reset = 0h]**

ADCRESULT11 is shown in [Figure 13-100](#) and described in [Table 13-94](#).

Return to the [Summary Table](#).

ADC Result 11 Register

**Figure 13-100. ADCRESULT11 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-94. ADCRESULT11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	<p>ADC Result 11</p> <p>16-bit ADC result. After the ADC completes a conversion of SOC11, the digital result is placed in this bit field.</p> <p>Reset type: SYSRSn</p>

**13.16.3.13 ADCRESULT12 Register (Offset = Ch) [reset = 0h]**

ADCRESULT12 is shown in [Figure 13-101](#) and described in [Table 13-95](#).

Return to the [Summary Table](#).

ADC Result 12 Register

**Figure 13-101. ADCRESULT12 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-95. ADCRESULT12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 12  16-bit ADC result. After the ADC completes a conversion of SOC12, the digital result is placed in this bit field. Reset type: SYSRSn

**13.16.3.14 ADCRESULT13 Register (Offset = Dh) [reset = 0h]**

ADCRESULT13 is shown in [Figure 13-102](#) and described in [Table 13-96](#).

Return to the [Summary Table](#).

ADC Result 13 Register

**Figure 13-102. ADCRESULT13 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-96. ADCRESULT13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	<p>ADC Result 13</p> <p>16-bit ADC result. After the ADC completes a conversion of SOC13, the digital result is placed in this bit field.</p> <p>Reset type: SYSRSn</p>

**13.16.3.15 ADCRESULT14 Register (Offset = Eh) [reset = 0h]**

ADCRESULT14 is shown in [Figure 13-103](#) and described in [Table 13-97](#).

Return to the [Summary Table](#).

ADC Result 14 Register

**Figure 13-103. ADCRESULT14 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-97. ADCRESULT14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 14  16-bit ADC result. After the ADC completes a conversion of SOC14, the digital result is placed in this bit field. Reset type: SYSRSn

**13.16.3.16 ADCRESULT15 Register (Offset = Fh) [reset = 0h]**

ADCRESULT15 is shown in [Figure 13-104](#) and described in [Table 13-98](#).

Return to the [Summary Table](#).

ADC Result 15 Register

**Figure 13-104. ADCRESULT15 Register**

15	14	13	12	11	10	9	8
RESULT							
R-0h							
7	6	5	4	3	2	1	0
RESULT							
R-0h							

**Table 13-98. ADCRESULT15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 15  16-bit ADC result. After the ADC completes a conversion of SOC15, the digital result is placed in this bit field. Reset type: SYSRSn

### 13.16.3.17 ADCPPB1RESULT Register (Offset = 10h) [reset = 0h]

ADCPPB1RESULT is shown in [Figure 13-105](#) and described in [Table 13-99](#).

Return to the [Summary Table](#).

ADC Post Processing Block 1 Result Register

**Figure 13-105. ADCPPB1RESULT 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
SIGN																PPBRESULT															
R-0h																R-0h															

**Table 13-99. ADCPPB1RESULT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	SIGN	R	0h	Sign Extended Bits. These bits reflect the same value as bit 16.  NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the SIGN bits extend down to bit 12, and all reflect the same value as bit 12. Reset type: SYSRSn
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result 1  The result of the offset/reference subtraction post conversion processing is stored in this register. If ADCINTFLG is polled in reading PPBRESULT, user needs to add a NOP instruction to ensure that post conversion processing is populated in this register.  NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0. Reset type: SYSRSn



**13.16.3.18 ADCPPB2RESULT Register (Offset = 12h) [reset = 0h]**

ADCPPB2RESULT is shown in [Figure 13-106](#) and described in [Table 13-100](#).

Return to the [Summary Table](#).

ADC Post Processing Block 2 Result Register

**Figure 13-106. ADCPPB2RESULT 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
SIGN																PPBRESULT															
R-0h																R-0h															

**Table 13-100. ADCPPB2RESULT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	SIGN	R	0h	<p>Sign Extended Bits. These bits reflect the same value as bit 16.</p> <p>NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the SIGN bits extend down to bit 12, and all reflect the same value as bit 12.</p> <p>Reset type: SYSRSn</p>
15-0	PPBRESULT	R	0h	<p>ADC Post Processing Block Result 2</p> <p>The result of the offset/reference subtraction post conversion processing is stored in this register. If ADCINTFLG is polled in reading PPBRESULT, user needs to add a NOP instruction to ensure that post conversion processing is populated in this register.</p> <p>NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0.</p> <p>Reset type: SYSRSn</p>

**13.16.3.19 ADCPPB3RESULT Register (Offset = 14h) [reset = 0h]**

ADCPPB3RESULT is shown in [Figure 13-107](#) and described in [Table 13-101](#).

Return to the [Summary Table](#).

ADC Post Processing Block 3 Result Register

**Figure 13-107. ADCPPB3RESULT 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
SIGN																PPBRESULT															
R-0h																R-0h															

**Table 13-101. ADCPPB3RESULT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	SIGN	R	0h	Sign Extended Bits. These bits reflect the same value as bit 16.  NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the SIGN bits extend down to bit 12, and all reflect the same value as bit 12. Reset type: SYSRSn
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result 3  The result of the offset/reference subtraction post conversion processing is stored in this register. If ADCINTFLG is polled in reading PPBRESULT, user needs to add a NOP instruction to ensure that post conversion processing is populated in this register.  NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0. Reset type: SYSRSn

**13.16.3.20 ADCPPB4RESULT Register (Offset = 16h) [reset = 0h]**

ADCPPB4RESULT is shown in [Figure 13-108](#) and described in [Table 13-102](#).

Return to the [Summary Table](#).

ADC Post Processing Block 4 Result Register

**Figure 13-108. ADCPPB4RESULT 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
SIGN																PPBRESULT															
R-0h																R-0h															

**Table 13-102. ADCPPB4RESULT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	SIGN	R	0h	<p>Sign Extended Bits. These bits reflect the same value as bit 16.</p> <p>NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the SIGN bits extend down to bit 12, and all reflect the same value as bit 12.</p> <p>Reset type: SYSRSn</p>
15-0	PPBRESULT	R	0h	<p>ADC Post Processing Block Result 4</p> <p>The result of the offset/reference subtraction post conversion processing is stored in this register. If ADCINTFLG is polled in reading PPBRESULT, user needs to add a NOP instruction to ensure that post conversion processing is populated in this register.</p> <p>NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0.</p> <p>Reset type: SYSRSn</p>

### 13.16.4 Register to Driverlib Function Mapping

**Table 13-103. ADC Registers to Driverlib Functions**

File	Driverlib Function
<b>CTL1</b>	
adc.h	ADC_setInterruptPulseMode
adc.h	ADC_enableConverter
adc.h	ADC_disableConverter
adc.h	ADC_isBusy
<b>CTL2</b>	
adc.h	ADC_setPrescaler
<b>BURSTCTL</b>	
adc.h	ADC_setBurstModeConfig
adc.h	ADC_enableBurstMode
adc.h	ADC_disableBurstMode
<b>INTFLG</b>	
adc.h	ADC_getInterruptStatus
adc.h	ADC_clearInterruptStatus
<b>INTFLGCLR</b>	
adc.h	ADC_clearInterruptStatus
<b>INTOVF</b>	
adc.h	ADC_getInterruptOverflowStatus
adc.h	ADC_clearInterruptOverflowStatus
<b>INTOVFCLR</b>	
adc.h	ADC_clearInterruptOverflowStatus
<b>INTSEL1N2</b>	
adc.h	ADC_enableInterrupt
adc.h	ADC_disableInterrupt
adc.h	ADC_setInterruptSource
adc.h	ADC_enableContinuousMode
adc.h	ADC_disableContinuousMode
<b>INTSEL3N4</b>	
-	See INTSEL1N2
<b>SOCPRCTL</b>	
adc.h	ADC_setSOCPriority
<b>INTSOCSEL1</b>	
adc.h	ADC_setInterruptSOCTrigger
<b>INTSOCSEL2</b>	
-	See INTSOCSEL1
<b>SOCFRC1</b>	
adc.h	ADC_forceSOC
<b>SOC0CTL</b>	
adc.h	ADC_setupSOC
<b>SOC1CTL</b>	
-	See SOC0CTL
<b>SOC2CTL</b>	
-	See SOC0CTL
<b>SOC3CTL</b>	
-	See SOC0CTL

**Table 13-103. ADC Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>SOC4CTL</b>	
-	See SOC0CTL
<b>SOC5CTL</b>	
-	See SOC0CTL
<b>SOC6CTL</b>	
-	See SOC0CTL
<b>SOC7CTL</b>	
-	See SOC0CTL
<b>SOC8CTL</b>	
-	See SOC0CTL
<b>SOC9CTL</b>	
-	See SOC0CTL
<b>SOC10CTL</b>	
-	See SOC0CTL
<b>SOC11CTL</b>	
-	See SOC0CTL
<b>SOC12CTL</b>	
-	See SOC0CTL
<b>SOC13CTL</b>	
-	See SOC0CTL
<b>SOC14CTL</b>	
-	See SOC0CTL
<b>SOC15CTL</b>	
-	See SOC0CTL
<b>EVTSTAT</b>	
adc.h	ADC_getPPBEventStatus
<b>EVTCLR</b>	
adc.h	ADC_clearPPBEventStatus
<b>EVTSEL</b>	
adc.h	ADC_enablePPBEvent
adc.h	ADC_disablePPBEvent
<b>EVTINTSEL</b>	
adc.h	ADC_enablePPBEventInterrupt
adc.h	ADC_disablePPBEventInterrupt
<b>OFFTRIM</b>	
adc.c	ADC_setVREF
<b>PPB1CONFIG</b>	
adc.h	ADC_setupPPB
adc.h	ADC_enablePPBTwosComplement
adc.h	ADC_disablePPBTwosComplement
<b>PPB1STAMP</b>	
adc.h	ADC_getPPBDelayTimeStamp
<b>PPB1OFFCAL</b>	
adc.h	ADC_setPPBCalibrationOffset
<b>PPB1OFFREF</b>	
adc.h	ADC_setPPBReferenceOffset

**Table 13-103. ADC Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>PPB1TRIPHI</b>	
adc.c	ADC_setPPBTripLimits
<b>PPB1TRIPLO</b>	
adc.c	ADC_setPPBTripLimits
<b>PPB2CONFIG</b>	
-	See PPB1CONFIG
<b>PPB2STAMP</b>	
-	See PPB1STAMP
<b>PPB2OFFCAL</b>	
-	See PPB1OFFCAL
<b>PPB2OFFREF</b>	
-	See PPB1OFFREF
<b>PPB2TRIPHI</b>	
-	See PPB1TRIPHI
<b>PPB2TRIPLO</b>	
-	See PPB1TRIPLO
<b>PPB3CONFIG</b>	
-	See PPB1CONFIG
<b>PPB3STAMP</b>	
-	See PPB1STAMP
<b>PPB3OFFCAL</b>	
-	See PPB1OFFCAL
<b>PPB3OFFREF</b>	
-	See PPB1OFFREF
<b>PPB3TRIPHI</b>	
-	See PPB1TRIPHI
<b>PPB3TRIPLO</b>	
-	See PPB1TRIPLO
<b>PPB4CONFIG</b>	
-	See PPB1CONFIG
<b>PPB4STAMP</b>	
-	See PPB1STAMP
<b>PPB4OFFCAL</b>	
-	See PPB1OFFCAL
<b>PPB4OFFREF</b>	
-	See PPB1OFFREF
<b>PPB4TRIPHI</b>	
-	See PPB1TRIPHI
<b>PPB4TRIPLO</b>	
-	See PPB1TRIPLO
<b>INTCYCLE</b>	
adc.h	ADC_setInterruptCycleOffset
<b>RESULT0</b>	
adc.h	ADC_readResult
<b>RESULT1</b>	
-	See RESULT0

**Table 13-103. ADC Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>RESULT2</b>	
-	See RESULT0
<b>RESULT3</b>	
-	See RESULT0
<b>RESULT4</b>	
-	See RESULT0
<b>RESULT5</b>	
-	See RESULT0
<b>RESULT6</b>	
-	See RESULT0
<b>RESULT7</b>	
-	See RESULT0
<b>RESULT8</b>	
-	See RESULT0
<b>RESULT9</b>	
-	See RESULT0
<b>RESULT10</b>	
-	See RESULT0
<b>RESULT11</b>	
-	See RESULT0
<b>RESULT12</b>	
-	See RESULT0
<b>RESULT13</b>	
-	See RESULT0
<b>RESULT14</b>	
-	See RESULT0
<b>RESULT15</b>	
-	See RESULT0
<b>PPB1RESULT</b>	
adc.h	ADC_readPPBResult
<b>PPB2RESULT</b>	
-	See PPB1RESULT
<b>PPB3RESULT</b>	
-	See PPB1RESULT
<b>PPB4RESULT</b>	
-	See PPB1RESULT

## Programmable Gain Amplifier (PGA)

---



---

The Programmable Gain Amplifier (PGA) is used to amplify an input voltage for the purpose of increasing the dynamic range of the downstream ADC and CMPSS modules.

Topic	Page
14.1 Programmable Gain Amplifier (PGA) Overview .....	1657
14.2 Linear Output Range .....	1658
14.3 Gain Modes .....	1658
14.4 External Filtering .....	1658
14.5 Error Calibration .....	1659
14.6 Ground Routing .....	1660
14.7 Enabling and Disabling the PGA Clock .....	1661
14.8 Lock Register .....	1661
14.9 Examples .....	1662
14.10 Analog Front End Integration .....	1662
14.11 PGA Registers.....	1665



## 14.1 Programmable Gain Amplifier (PGA) Overview

The integrated PGA helps to reduce cost and design effort for many control applications that traditionally require external, standalone amplifiers. On-chip integration ensures that the PGA is compatible with the downstream ADC and CMPSS modules. Software selectable gain and filter settings make the PGA adaptable to various performance needs.

### 14.1.1 Features

Features available to PGA modules are:

- Four programmable gain modes: 3x, 6x, 12x, 24x
- Internally powered by VDDA and VSSA
- Hardware-based trims to reduce offset and gain errors
- Support for Kelvin ground connections using PGA\_GND pin
- Embedded series resistors for RC filtering

### 14.1.2 Block Diagram

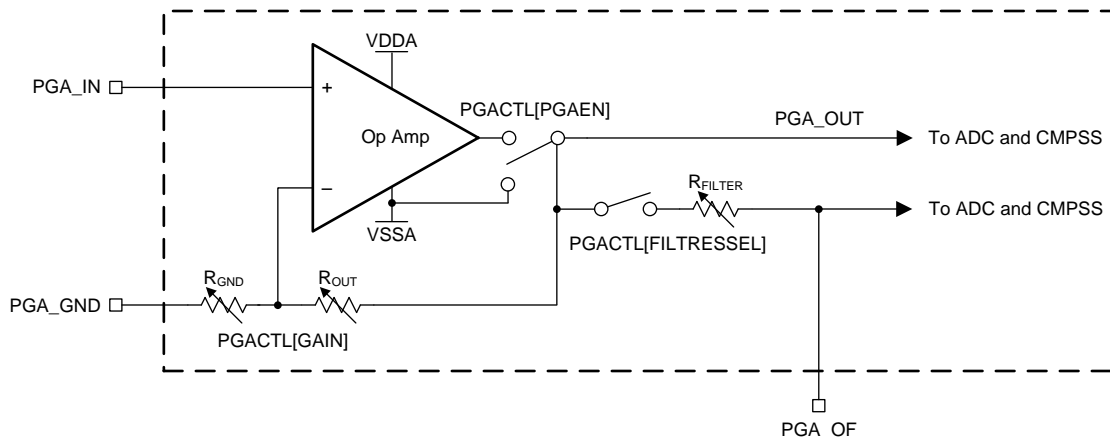
The active component in the PGA is an embedded operational amplifier (op-amp) that is configured as a non-inverting amplifier with internal feedback resistors. These internal feedback resistor values are paired to produce software selectable voltage gains.

Three PGA signals are available at the device pins:

- PGA\_IN is the positive input to the PGA op-amp. The signal applied to this pin will be amplified by the PGA.
- PGA\_GND is the Kelvin ground reference for the PGA\_IN signal. Ideally, the PGA\_GND reference is equal to VSSA, however the PGA can tolerate small voltage offsets from VSSA. See the device data manual for more information.
- PGA\_OF supports op-amp output filtering with RC components. The filtered signal is available for sampling and monitoring by internal ADC and CMPSS modules.

PGA\_OUT is an internal signal at the op-amp output. It is available for sampling and monitoring by the internal ADC and CMPSS modules.

**Figure 14-1. PGA Block Diagram**



## 14.2 Linear Output Range

The absolute output range of the PGA is bounded by the analog VDDA and VSSA supplies – the PGA cannot produce output voltages greater than VDDA or less than VSSA.

Although the PGA can produce full-scale output across the absolute voltage range of VSSA to VDDA, the amplifier output is only linear within a subset of the absolute range. This reduced range is referred to as the linear output range.

The PGA performance specifications in the device data manual only apply to the linear output range. For best performance, the input signal should be conditioned in such a way that the PGA stays within the linear output range during normal system operation.

---

**NOTE:** The voltage input range required to operate the PGA in the linear output range is unique for each gain mode. See the device data manual for the linear output range.

---

## 14.3 Gain Modes

Gain modes of 3x, 6x, 12x, and 24x are software-selectable via the PGACTL[GAIN] register field. The gain of the PGA is determined by a preset ratio between resistors  $R_{OUT}$  and  $R_{GND}$ :

$$\text{Gain} = 1 + \frac{R_{OUT}}{R_{GND}}$$

The ideal target values for the gain resistors by mode are shown in [Table 14-1](#).

**Table 14-1. Ideal Gain Resistor Values**

Gain Mode	Ideal $R_{OUT}$	Ideal $R_{GND}$
3x	20k $\Omega$	10k $\Omega$
6x	25k $\Omega$	5k $\Omega$
12x	27.5k $\Omega$	2.5k $\Omega$
24x	28.75k $\Omega$	1.25k $\Omega$

Changing the gain mode during normal operation is allowed, but a minimum configuration settling time should be observed when doing so. See the device data manual for the gain switch settling time.

## 14.4 External Filtering

The PGA output may be routed to a pin through an embedded series resistor for the purpose of low-pass filtering the amplified signal. The filter resistance is software selectable via the PGACTL[FILTRESSEL] register field. The default selection of PGACTL[FILTRESSEL]=0 will disable the filter path.

The cutoff frequency can be estimated using the standard low-pass RC equation of:

$$f_C = \frac{1}{2\pi RC}$$

Each gain mode requires a minimum amount of series resistance when filtering is enabled. The values are shown in [Table 14-2](#).

**Table 14-2. Minimum Filter Resistance**

Gain Mode	Minimum $R_{FILTER}$
3x	50 $\Omega$
6x	50 $\Omega$
12x	80 $\Omega$
24x	100 $\Omega$

The choice of  $C_{FILTER}$  value may also influence the ADC sampling performance. See [Section 14.10.1.2](#) for more information.

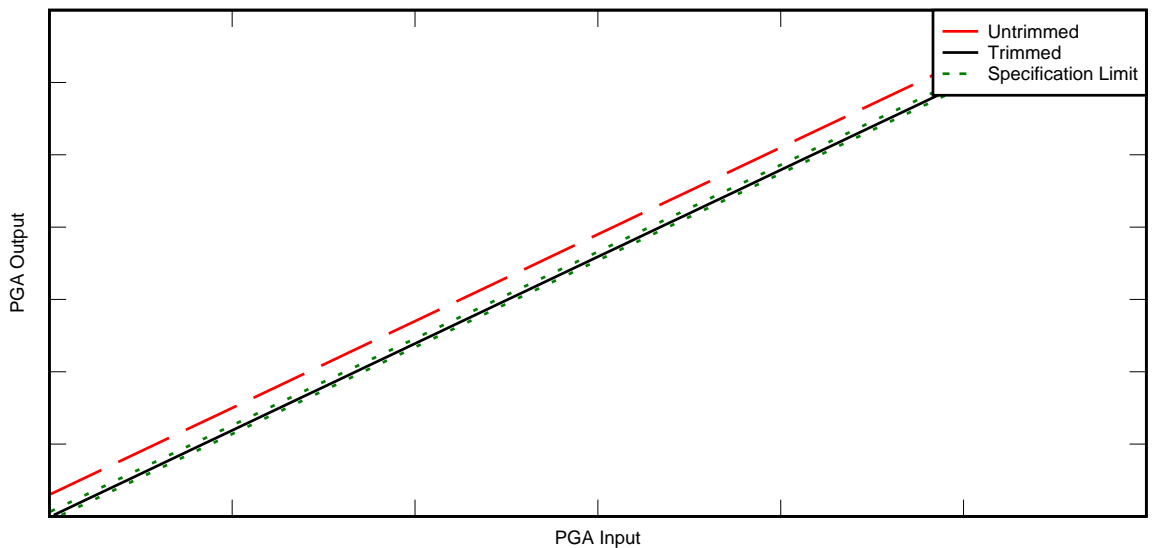
## 14.5 Error Calibration

Inherent offset and gain errors can be reduced through the use of built-in hardware trim circuitry. Factory-generated values are written to the trim registers by calling the `Device_cal()` function that is located in TI reserved OTP.

### 14.5.1 Offset Error

The offset error appears as a constant DC offset across the PGA output range. The hardware trim will reduce the offset error so that it falls within data manual specifications.

**Figure 14-2. PGA Offset Trim**

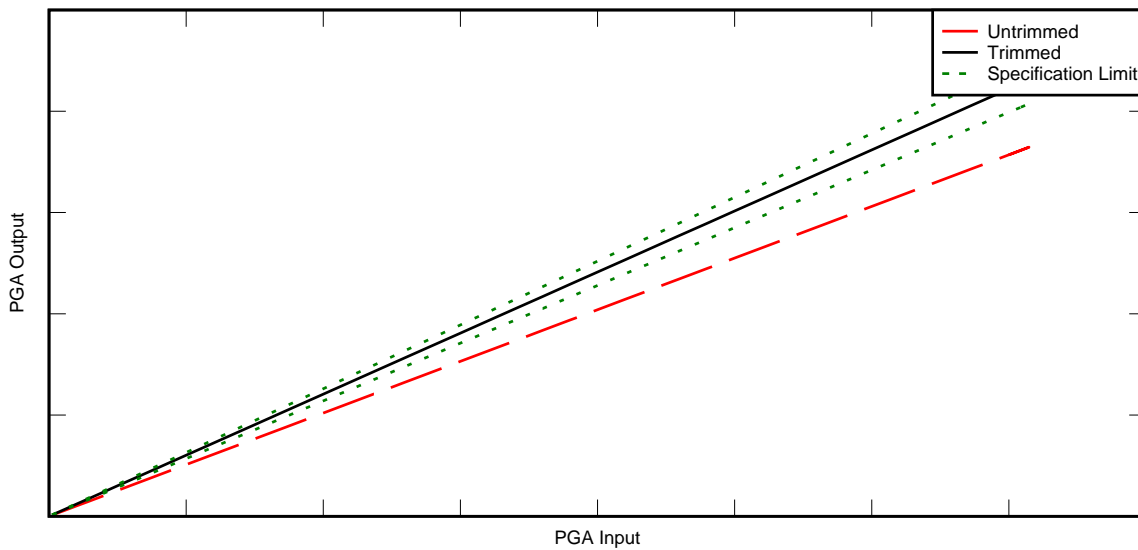


D001

### 14.5.2 Gain Error

After the offset error has been removed, the remaining error, gain error appears as a scaled error that increases in magnitude with increasing PGA output voltage. The hardware trim will target the gain performance so that it falls within data manual specifications.

Figure 14-3. PGA Gain Trim

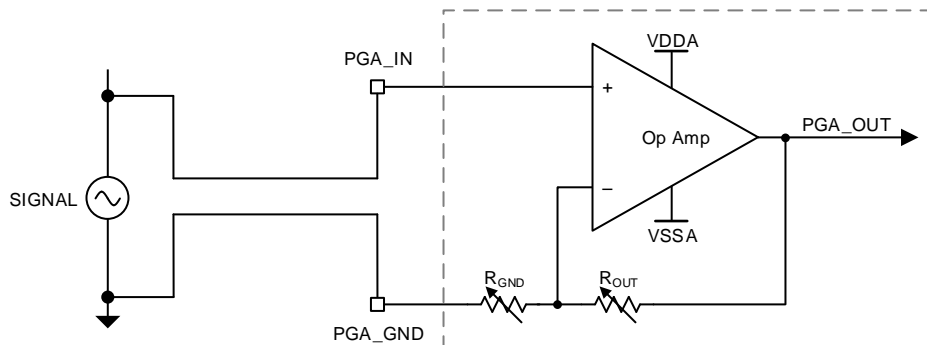


D002

### 14.6 Ground Routing

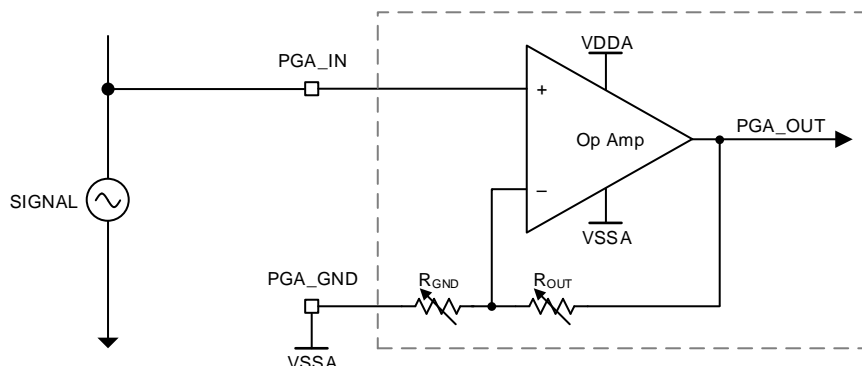
PGA\_GND should be connected to the source signal ground reference when possible. Routing the PGA\_IN and PGA\_GND signals in close proximity (especially in parallel) may also help to reduce sensitivity to external noise.

Figure 14-4. PGA\_GND to Remote Ground



PGA\_GND may also be connected to VSSA if the source signal ground reference is not available.

Figure 14-5. PGA\_GND to VSSA



PGA\_GND should not be biased with an intentional DC offset with respect to VSSA.

## **14.7 Enabling and Disabling the PGA Clock**

If the clock to the PGA is disabled while the PGA is outputting a voltage, the output voltage remains unaffected but the PGA registers will no longer be updated with register writes. Enabling the clock resumes register writes.

## **14.8 Lock Register**

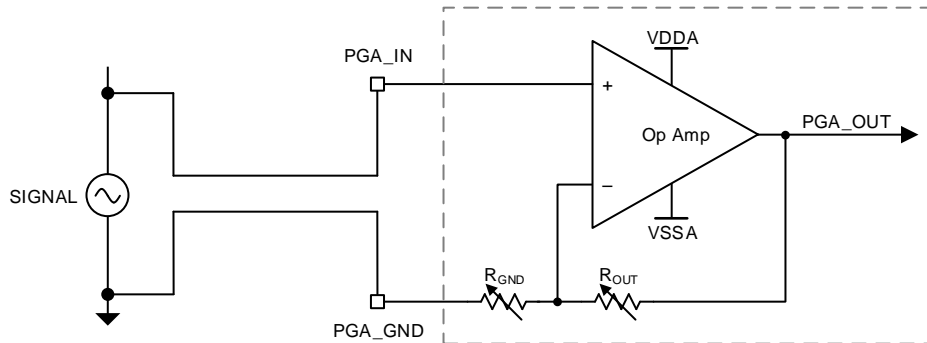
The PGALOCK register provides the ability to block writes to certain PGA configuration registers. Locking register writes can prevent spurious or malicious code from modifying critical register settings. Once a register has been locked via the PGALOCK register, only a module- or device-level reset can restore write functionality.

## 14.9 Examples

### 14.9.1 Direct Amplifier

Given a small positive signal, the PGA can be used to amplify the signal to increase the dynamic range of ADC sampling and comparator trip monitoring. For example, an input signal with a valid range between 0.25V and 0.75V can be amplified in 3x mode to produce an output signal between 0.75V and 2.25V.

**Figure 14-6. PGA Amplifier Example**



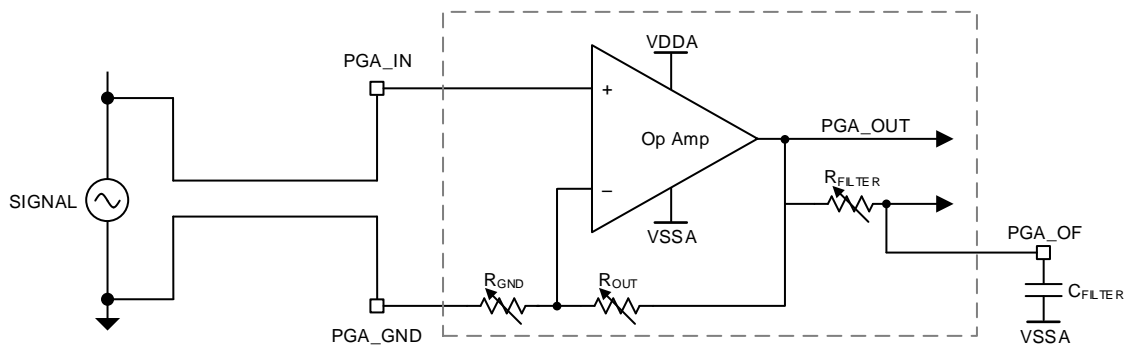
The amplified output voltage is calculated as:

$$V_{\text{PGA\_OUT}} = \left( 1 + \frac{R_{\text{OUT}}}{R_{\text{GND}}} \right) \times V_{\text{PGA\_IN}}$$

### 14.9.2 RC Filter

If low-pass filtering is desired, an external capacitor can be added to the PGA\_OF pin in the following topology.

**Figure 14-7. PGA Filter Example**



The amplified voltage is calculated as:

$$V_{\text{PGA\_OUT}} = \left( 1 + \frac{R_{\text{OUT}}}{R_{\text{GND}}} \right) \times V_{\text{PGA\_IN}}$$

The filter cutoff frequency is estimated as:

$$f_{\text{CUTOFF}} = \frac{1}{2\pi R_{\text{FILTER}} C_{\text{FILTER}}}$$

## 14.10 Analog Front End Integration

The PGAs operate in concert with the other embedded analog modules (ADC, CMPSS, Buffered DAC) as an analog front end system.

### 14.10.1 ADC

In its simplest application, the PGA amplifies small input signals in order to increase the ADC dynamic range. The PGA also provides the additional benefit of buffering input signals from the ADC sample and hold capacitor, which further reduces sampling error.

Both the filtered and unfiltered PGA output paths are available for ADC sampling. Minimum ADC acquisition windows are recommended when sampling the PGA paths.

#### 14.10.1.1 Unfiltered Acquisition Window

The device data manual provides a minimum estimated ADC acquisition window for sampling the PGA\_OUT signal with one ADC. This estimated value should provide sampling accuracy close to the specified performance parameters of the ADC. A longer acquisition window may be used for better performance.

#### 14.10.1.2 Filtered Acquisition Window

The minimum ADC acquisition window for sampling the PGA\_OF filtered signal with one ADC will vary based on the values of  $R_{\text{FILTER}}$  and  $C_{\text{FILTER}}$ . To ensure good performance, choose a  $C_{\text{FILTER}}$  capacitor that is large enough to satisfy most of the ADC sample and hold capacitor ( $C_h$ ) charge requirements by itself. The  $C_{\text{FILTER}}$  value can be sized based on the acceptable amount of ADC sampling error ( $\text{LSB}_{\text{Err}}$ ):

$$C_{\text{FILTER}} = C_h \times 4096 / \text{LSB}_{\text{Err}}$$

See the *Analog-to-Digital Converter (ADC)* chapter to determine the optimal ADC acquisition window, where the ADC source resistance ( $R_s$ ) is equal to  $R_{\text{FILTER}}$ .

### 14.10.2 CMPSS

The PGA output can be monitored by a CMPSS module for trips above or below a reference voltage. Up to two independent reference thresholds per CMPSS can be used for trip detection.

Both the filtered and unfiltered PGA output paths are available for CMPSS trip monitoring.

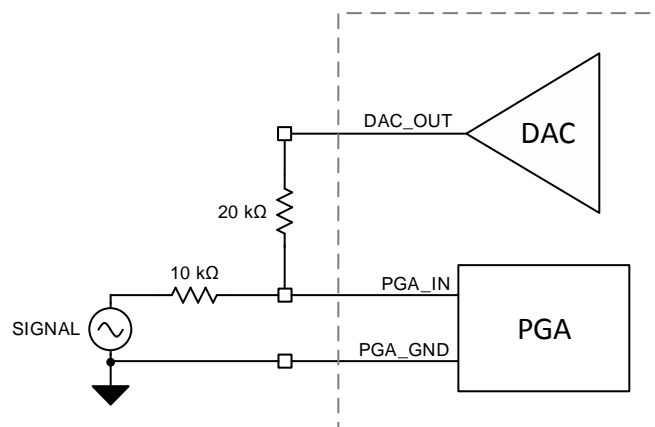
See the *Comparator Subsystem (CMPSS)* chapter for CMPSS usage information.

### 14.10.3 Buffered DAC

As a best practice, the PGA input signal should be conditioned so that the PGA output is centered within its linear range. The input signal may require some combination of offset and attenuation in order to achieve this goal.

For example, an external resistor divider can attenuate the input signal while the embedded buffered DAC can provide a positive voltage offset.

Figure 14-8. Buffered DAC Offset



With the topology shown above, the voltage seen at the PGA\_IN pin can be calculated as follows:

$$V_{PGA\_IN} = \frac{20\text{ k}\Omega \times (V_{SIGNAL} - V_{DAC\_OUT})}{10\text{ k}\Omega + 20\text{ k}\Omega} + V_{DAC\_OUT}$$

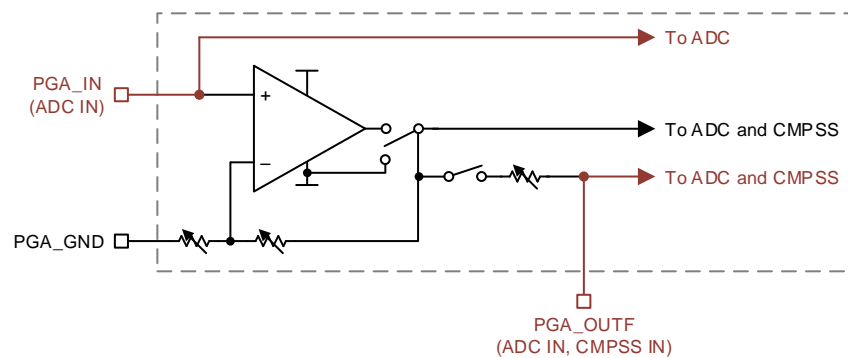
Supplying a  $V_{DAC\_OUT}$  of 1.65V would transform a bipolar  $V_{SIGNAL}$  range of -0.5V to +0.5V into a  $V_{PGA\_IN}$  range of 0.22V to 0.88V, which is ideal for the 3x gain mode.

See the *Buffered Digital to Analog Converter (DAC)* chapter for buffered DAC usage information.

### 14.10.4 Alternate Functions

Each PGA has up to three device terminals for operation: ground, input, and output filter. Alternate paths at the device level allow the input and output filter terminals to take on alternate functions. This can be especially valuable if the respective PGA resource is not used for an application. The ground terminal should be tied to VSSA if the PGA is not used.

**Figure 14-9. PGA Pin Alternate Functions**



See the *Analog Subsystem* chapter for more information.



## 14.11 PGA Registers

### 14.11.1 PGA Base Addresses

**Table 14-3. I2C Base Address Table**

Device Registers	Register Name	Start Address	End Address
Pga1Regs	PGA_REGS	0000 5B00	0000 5B0F
Pga2Regs	PGA_REGS	0000 5B10	0000 5B1F
Pga3Regs	PGA_REGS	0000 5B20	0000 5B2F
Pga4Regs	PGA_REGS	0000 5B30	0000 5B3F
Pga5Regs	PGA_REGS	0000 5B40	0000 5B4F
Pga6Regs	PGA_REGS	0000 5B50	0000 5B5F
Pga7Regs	PGA_REGS	0000 5B60	0000 5B6F

### 14.11.2 PGA\_REGS Registers

Table 14-4 lists the PGA\_REGS registers. All register offset addresses not listed in Table 14-4 should be considered as reserved locations and the register contents should not be modified.

**Table 14-4. PGA\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	PGACTL	PGA Control Register	EALLOW	<a href="#">Go</a>
2h	PGALOCK	PGA Lock Register	EALLOW	<a href="#">Go</a>
4h	PGAGAIN3TRIM	PGA Gain Trim Register for a gain setting of 3	EALLOW	<a href="#">Go</a>
5h	PGAGAIN6TRIM	PGA Gain Trim Register for a gain setting of 6	EALLOW	<a href="#">Go</a>
6h	PGAGAIN12TRIM	PGA Gain Trim Register for a gain setting of 12	EALLOW	<a href="#">Go</a>
7h	PGAGAIN24TRIM	PGA Gain Trim Register for a gain setting of 24	EALLOW	<a href="#">Go</a>
8h	PGATYPE	PGA Type Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 14-5 shows the codes that are used for access types in this section.

**Table 14-5. PGA\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
WOnce	W Once	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**14.11.2.1 PGACTL Register (Offset = 0h) [reset = 0h]**

 PGACTL is shown in [Figure 14-10](#) and described in [Table 14-6](#).

 Return to the [Summary Table](#).

PGA Control Register

**Figure 14-10. PGACTL 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							
7	6	5	4	3	2	1	0
GAIN			FILTRESSEL			PGAEN	
R/W-0h			R/W-0h			R/W-0h	

**Table 14-6. PGACTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R/W	0h	Reserved
7-5	GAIN	R/W	0h	PGA Gain. 000 x 3 001 x 6 010 x 12 011 x 24 Reset type: SYSRSn
4-1	FILTRESSEL	R/W	0h	0000 Filter disabled (default) 0001 Filter Resistance 200 Ohm 0010 Filter Resistance 160 Ohm 0011 Filter Resistance 130 Ohm 0100 Filter Resistance 100 Ohm 0101 Filter Resistance 80 Ohm 0110 Filter Resistance 50 Ohm Reset type: SYSRSn
0	PGAEN	R/W	0h	PGA Enable. 0 PGA is disabled and powered down. 1 PGA is enabled. Reset type: SYSRSn

**14.11.2.2 PGALOCK Register (Offset = 2h) [reset = 0h]**

PGALOCK is shown in [Figure 14-11](#) and described in [Table 14-7](#).

Return to the [Summary Table](#).

PGA Lock Register

**Figure 14-11. PGALOCK Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	RESERVED	PGAGAIN24TRIM	PGAGAIN12TRIM	PGAGAIN6TRIM	PGAGAIN3TRIM	RESERVED	PGACTL
R-0h		WSonce-0h	WSonce-0h	WSonce-0h	WSonce-0h	WSonce-0h	

**Table 14-7. PGALOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	RESERVED	WSonce	0h	Reserved
6	RESERVED	R	0h	Reserved
5	PGAGAIN24TRIM	WSonce	0h	0 Writes to PGAGAIN24TRIM are enabled. 1 Writes to PGAGAIN24TRIM are disabled. Reset type: SYSRSn
4	PGAGAIN12TRIM	WSonce	0h	0 Writes to PGAGAIN12TRIM are enabled. 1 Writes to PGAGAIN12TRIM are disabled. Reset type: SYSRSn
3	PGAGAIN6TRIM	WSonce	0h	0 Writes to PGAGAIN6TRIM are enabled. 1 Writes to PGAGAIN6TRIM are disabled. Reset type: SYSRSn
2	PGAGAIN3TRIM	WSonce	0h	0 Writes to PGAGAIN3TRIM are enabled. 1 Writes to PGAGAIN3TRIM are disabled. Reset type: SYSRSn
1	RESERVED	WSonce	0h	Reserved
0	PGACTL	WSonce	0h	0 Writes to PGACTL are enabled. 1 Writes to PGACTL are disabled. Reset type: SYSRSn

### 14.11.2.3 PGAGAIN3TRIM Register (Offset = 4h) [reset = 0h]

PGAGAIN3TRIM is shown in [Figure 14-12](#) and described in [Table 14-8](#).

Return to the [Summary Table](#).

PGA Gain Trim Register for a gain setting of 3

**Figure 14-12. PGAGAIN3TRIM Register**

15	14	13	12	11	10	9	8
OFFSETTRIM							
R/W-0h							
7	6	5	4	3	2	1	0
GAINTRIM							
R/W-0h							

**Table 14-8. PGAGAIN3TRIM Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	OFFSETTRIM	R/W	0h	Offset TRIM value, when Gain setting is 3 Reset type: SYSRSn
7-0	GAINTRIM	R/W	0h	Gain TRIM value, when gain setting is 3 Reset type: SYSRSn

#### 14.11.2.4 PGAGAIN6TRIM Register (Offset = 5h) [reset = 0h]

PGAGAIN6TRIM is shown in [Figure 14-13](#) and described in [Table 14-9](#).

Return to the [Summary Table](#).

PGA Gain Trim Register for a gain setting of 6

**Figure 14-13. PGAGAIN6TRIM Register**

15	14	13	12	11	10	9	8
OFFSETTRIM							
R/W-0h							
7	6	5	4	3	2	1	0
GAINTRIM							
R/W-0h							

**Table 14-9. PGAGAIN6TRIM Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	OFFSETTRIM	R/W	0h	Offset TRIM value, when Gain setting is 6 Reset type: SYSRSn
7-0	GAINTRIM	R/W	0h	Gain TRIM value, when gain setting is 6 Reset type: SYSRSn

**14.11.2.5 PGAGAIN12TRIM Register (Offset = 6h) [reset = 0h]**

PGAGAIN12TRIM is shown in [Figure 14-14](#) and described in [Table 14-10](#).

Return to the [Summary Table](#).

PGA Gain Trim Register for a gain setting of 12

**Figure 14-14. PGAGAIN12TRIM Register**

15	14	13	12	11	10	9	8
OFFSETTRIM							
R/W-0h							
7	6	5	4	3	2	1	0
GAINTRIM							
R/W-0h							

**Table 14-10. PGAGAIN12TRIM Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	OFFSETTRIM	R/W	0h	Offset TRIM value, when Gain setting is 12 Reset type: SYSRSn
7-0	GAINTRIM	R/W	0h	Gain TRIM value, when gain setting is 12 Reset type: SYSRSn

#### 14.11.2.6 PGAGAIN24TRIM Register (Offset = 7h) [reset = 0h]

PGAGAIN24TRIM is shown in [Figure 14-15](#) and described in [Table 14-11](#).

Return to the [Summary Table](#).

PGA Gain Trim Register for a gain setting of 24

**Figure 14-15. PGAGAIN24TRIM Register**

15	14	13	12	11	10	9	8
OFFSETTRIM							
R/W-0h							
7	6	5	4	3	2	1	0
GAINTRIM							
R/W-0h							

**Table 14-11. PGAGAIN24TRIM Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	OFFSETTRIM	R/W	0h	Offset TRIM value, when Gain setting is 24 Reset type: SYSRSn
7-0	GAINTRIM	R/W	0h	Gain TRIM value, when gain setting is 24 Reset type: SYSRSn



### 14.11.2.7 PGATYPE Register (Offset = 8h) [reset = 0h]

PGATYPE is shown in [Figure 14-16](#) and described in [Table 14-12](#).

Return to the [Summary Table](#).

PGA Type Register

**Figure 14-16. PGATYPE Register**

15	14	13	12	11	10	9	8
TYPE							
R-0h							
7	6	5	4	3	2	1	0
REV							
R-0h							

**Table 14-12. PGATYPE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	TYPE	R	0h	PGA Type. Reset type: SYSRSn
7-0	REV	R	0h	PGA Revision. Reset type: SYSRSn

## ***Buffered Digital-to-Analog Converter (DAC)***

---

---

The buffered digital-to-analog converter (DAC) is an analog module that can output a programmable, arbitrary reference voltage.

<b>Topic</b>	<b>Page</b>
<b>15.1 Introduction .....</b>	<b>1675</b>
<b>15.2 Using the DAC .....</b>	<b>1675</b>
<b>15.3 Lock Registers .....</b>	<b>1677</b>
<b>15.4 DAC Registers .....</b>	<b>1678</b>

## 15.1 Introduction

The buffered DAC module consists of an internal 12-bit DAC and an analog output buffer that is capable of driving an external load. For driving even higher loads than typical, a trade-off can be made between load size and output voltage swing. For the load conditions of the buffered DAC, see the device-specific data manual. The buffered DAC is a general-purpose DAC that can be used to generate a DC voltage in addition to AC waveforms such as sine waves, square waves, triangle waves and so forth. Software writes to the DAC value register can take effect immediately or can be synchronized with EPWMSYNCPER events.

### 15.1.1 Features

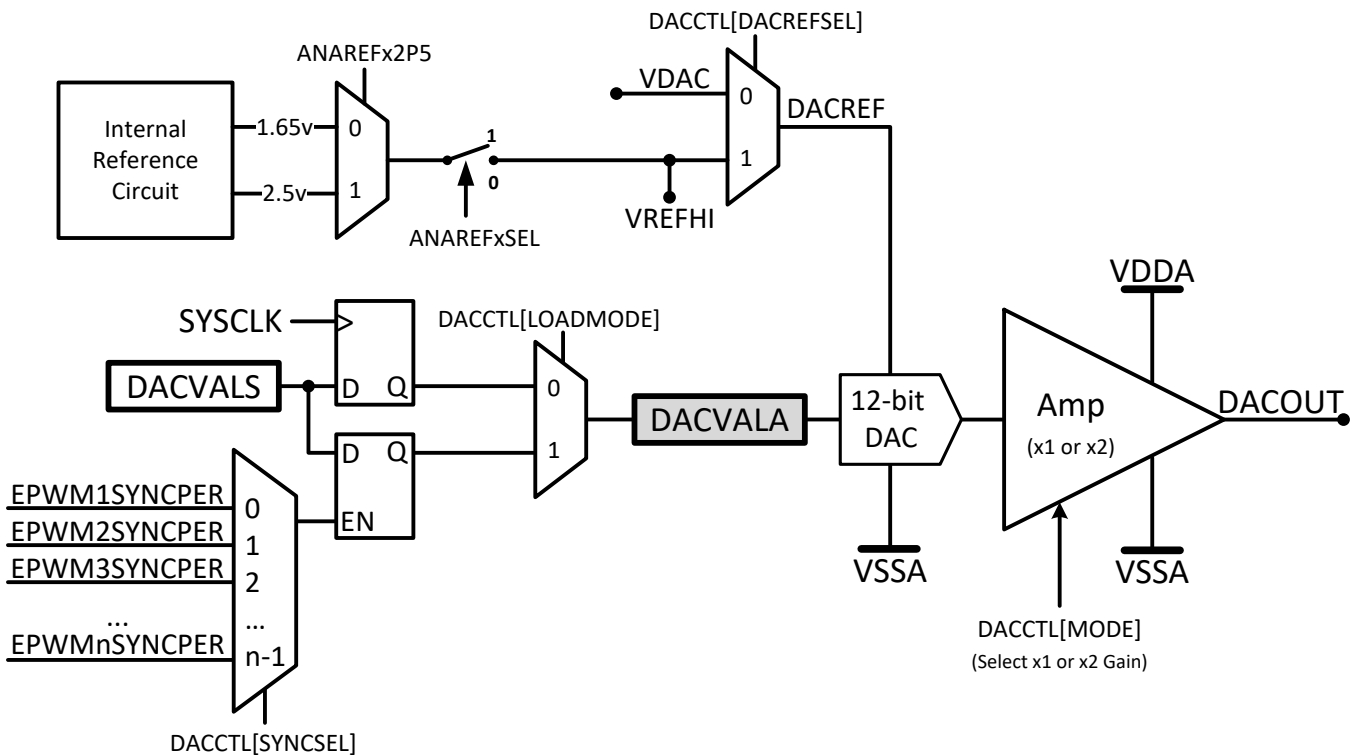
Each buffered DAC has the following features:

- 12-bit programmable internal DAC
- Selectable reference voltage source
- x1 and x2 gain modes when using internal VREFHI
- Ability to synchronize with EPWMSYNCPER

### 15.1.2 Block Diagram

The block diagram for the buffered DAC is shown in Figure 15-1.

Figure 15-1. DAC Module Block Diagram



## 15.2 Using the DAC

The internal DAC's reference voltage source, **DACREF**, is selectable between **VDAC** and **VREFHI**. The x2 gain mode is only available when **VREFHI** is set as **DACREF** and internal reference mode is used (see the *Analog Subsystem* chapter on how to switch to internal reference mode). Even though the buffered DAC has an x2 gain mode, the maximum output voltage from the buffered DAC will not be greater than **VDDA**. Table 15-1 lists the gain mode combinations supported by the buffered DAC. In this table, x = A or B, X = Don't Care, **VDAC/VREFHI** = 2.5V, **VDDA** = 3.3V, and **DACVAL** = 4095.

**Table 15-1. DAC Supported Gain Mode Combinations**

DACREFSEL	ANAREF <sub>x</sub> SEL	ANAREF <sub>x2P5</sub>	REF SOURCE	REF VOLTAGE (V)	MODE	MAXIMUM DAC OUTPUT (V)	SUPPORT STATUS
0	X	X	EXTERNAL	VDAC	0	2.5	SUPPORTED
0	X	X	EXTERNAL	VDAC	1	2.5	NOT SUPPORTED
1	0	0	INTERNAL	1.65	0	1.65	NOT SUPPORTED
1	0	0	INTERNAL	1.65	1	3.3	SUPPORTED
1	0	1	INTERNAL	2.5	0	2.5	SUPPORTED
1	0	1	INTERNAL	2.5	1	3.3	NOT SUPPORTED
1	1	X	EXTERNAL	VREFHI	0	2.5	SUPPORTED
1	1	X	EXTERNAL	VREFHI	1	2.5	NOT SUPPORTED

Two sets of DACVAL registers, DACVALA and DACVALS, are present in the buffered DAC module. DACVALA is a read-only register that actively controls the buffered DAC value. DACVALS is a writable shadow register that loads into DACVALA either immediately or synchronized with the next EPWMSYNCPER event. If the clock to the buffered DAC is disabled while the buffered DAC is outputting a voltage, the output voltage remains unaffected, but DACVALA and DACVALS will no longer be updated with register writes. Enabling the clock to the buffered DAC restores it to the state before the clock was disabled.

The ideal output of the internal DAC can be calculated as shown in [Equation 1](#).

(1)

$$DACOUT = \frac{DACVALA * DACREF}{4096}$$

The buffered DAC can be used to level-shift the PGA's input signal (see the PGA section for details). The output buffer of the buffered DAC may exhibit non-linear behavior near the supply rails (VDDA/VSSA). To determine the linear range of the buffered DAC, see the device-specific data manual.

### 15.2.1 Initialization Sequence

1. Enable the buffered DAC clock.
2. Set DACREF with DACREFSEL.
3. Power up the buffered DAC with DACOUTEN.
4. Wait for the power-up time to elapse before outputting a voltage. To determine the power-up time of the buffered DAC, see the device-specific data manual.
5. For predictable behavior of the buffered DAC, consecutive writes to DACVALS should be spaced apart according to the settling time of the buffered DAC. To determine the settling time of the buffered DAC, see the device-specific data manual.

### 15.2.2 DAC Offset Adjustment

Zero offset error is defined as the difference between the voltage at midcode (2048) and 1.25v (for 2.5v reference voltage). DAC offset error is calibrated at 2.5v reference voltage and loaded into the DAC offset trim register as part of the Device\_cal() function. If the DAC is used at any reference voltage other than 2.5v, the offset trim must be adjusted to ensure the offset error performance stays within the device-specific data manual limits. The DAC offset register is a 16-bit register that contains the 8-bit signed offset trim in the lower half of the register. Use the function call DAC\_tuneOffsetTrim() found in C2000Ware to adjust the offset.

### 15.2.3 EPWMSYNCPER Signal

EPWMSYNCPER comes from the Time-Base submodule of the EPWM. For a detailed description of how this signal is generated, refer to the Time-Base submodule chapter of the EPWM.

The EPWMSYNCPER signal that loads DACVALA when DACCTL [LOADMODE] = 1 is a level trigger load. If TBCTR and TBPRD of the EPWM are both 0, EPWMSYNCPER will be held at level high and DACVALA will be loaded immediately from DACVALS irrespective of the value of DACCTL [LOADMODE]. Due to this, it is recommended to configure the EPWM first before setting DACCTL [LOADMODE] to 1.

---

**NOTE:** The name of the sync signal that the GPDAC receives from the EPWM has been updated from PWMSYNC to EPWMSYNCPER (SYNCPER/PWMSYNCPER/EPWMxSYNCPER) to avoid confusion with the other EPWM sync signals EPWMSYNCI and EPWMSYNCO. For a description of what these signals are, see the EPWM chapter.

---

## 15.3 Lock Registers

A DACLOCK register is provided to prevent spurious writes from modifying the DACCTL, DACVALS, and DACOUTEN registers. Once a register is protected through DACLOCK, write access will be locked out until the device is reset.

## 15.4 DAC Registers

This section describes the Buffered Digital to Analog Converter registers.

### 15.4.1 DAC Base Addresses

**Table 15-2. DAC Base Address Table**

Device Registers	Register Name	Start Address	End Address
DacaRegs	DAC_REGS	0x0000_5C00	0x0000_5C0F
DacbRegs	DAC_REGS	0x0000_5C10	0x0000_5C1F

### 15.4.2 DAC\_REGS Registers

Table 15-3 lists the DAC\_REGS registers. All register offset addresses not listed in Table 15-3 should be considered as reserved locations and the register contents should not be modified.

**Table 15-3. DAC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	DACREV	DAC Revision Register		<a href="#">Go</a>
1h	DACCTL	DAC Control Register	EALLOW	<a href="#">Go</a>
2h	DACVALA	DAC Value Register - Active		<a href="#">Go</a>
3h	DACVALS	DAC Value Register - Shadow		<a href="#">Go</a>
4h	DACOUTEN	DAC Output Enable Register	EALLOW	<a href="#">Go</a>
5h	DACLOCK	DAC Lock Register	EALLOW	<a href="#">Go</a>
6h	DACTRIM	DAC Trim Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 15-4 shows the codes that are used for access types in this section.

**Table 15-4. DAC\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R -0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
WOnce	W Once	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 15.4.2.1 DACREV Register (Offset = 0h) [reset = 0h]

DACREV is shown in [Figure 15-2](#) and described in [Table 15-5](#).

Return to the [Summary Table](#).

DAC Revision Register

**Figure 15-2. DACREV Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
REV							
R-0h							

**Table 15-5. DACREV Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	REV	R	0h	DAC Revision Reset type: SYSRSn



### 15.4.2.2 DACCTL Register (Offset = 1h) [reset = 0h]

DACCTL is shown in [Figure 15-3](#) and described in [Table 15-6](#).

Return to the [Summary Table](#).

DAC Control Register

**Figure 15-3. DACCTL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
SYNCSEL				RESERVED	LOADMODE	MODE	DACREFSEL
R/W-0h				R-0h	R/W-0h	R/W-0h	R/W-0h

**Table 15-6. DACCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-4	SYNCSEL	R/W	0h	DAC EPWMSYNCPER select. Determines which EPWMSYNCPER signal will update the DACVALA register.  Where n represents the maximum number of EPWMSYNCPER signals available on the device:  0 EPWM1SYNCPER 1 EPWM2SYNCPER 2 EPWM3SYNCPER ... n-1 EPWMnSYNCPER Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	LOADMODE	R/W	0h	DACVALA load mode. Determines when the DACVALA register is updated with the value from DACVALS.  0 Load on next SYSCLK 1 Load on next EPWMSYNCPER specified by SYNCSEL Reset type: SYSRSn
1	MODE	R/W	0h	DAC gain mode select. Selects the gain mode for the buffered output. The MODE value is only used when DACREFSEL=1 and internal ADC reference mode is selected.  0 Gain is 1 1 Gain is 2 Reset type: SYSRSn
0	DACREFSEL	R/W	0h	DAC reference select. Selects which voltage references are used by the DAC.  0 VDAC/VSSA are the reference voltages 1 ADC VREFHI/VSSA are the reference voltages Reset type: SYSRSn

### 15.4.2.3 DACVALA Register (Offset = 2h) [reset = 0h]

DACVALA is shown in [Figure 15-4](#) and described in [Table 15-7](#).

Return to the [Summary Table](#).

DAC Value Register - Active

**Figure 15-4. DACVALA Register**

15	14	13	12	11	10	9	8
RESERVED				DACVALA			
R-0h				R-0h			
7	6	5	4	3	2	1	0
DACVALA							
R-0h							

**Table 15-7. DACVALA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVALA	R	0h	Active output code currently driven by the DAC Reset type: SYSRSn

**15.4.2.4 DACVALS Register (Offset = 3h) [reset = 0h]**

DACVALS is shown in [Figure 15-5](#) and described in [Table 15-8](#).

Return to the [Summary Table](#).

DAC Value Register - Shadow

**Figure 15-5. DACVALS Register**

15	14	13	12	11	10	9	8
RESERVED				DACVALS			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
DACVALS							
R/W-0h							

**Table 15-8. DACVALS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVALS	R/W	0h	Shadow output code to be loaded into DACVALA Reset type: SYSRSn

### 15.4.2.5 DACOUTEN Register (Offset = 4h) [reset = 0h]

DACOUTEN is shown in [Figure 15-6](#) and described in [Table 15-9](#).

Return to the [Summary Table](#).

DAC Output Enable Register

**Figure 15-6. DACOUTEN Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							DACOUTEN
R-0h							R/W-0h

**Table 15-9. DACOUTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	DACOUTEN	R/W	0h	DAC output enable 0 DAC output is disabled 1 DAC output is enabled Reset type: SYSRSn

**15.4.2.6 DACLOCK Register (Offset = 5h) [reset = 0h]**

DACLOCK is shown in [Figure 15-7](#) and described in [Table 15-10](#).

Return to the [Summary Table](#).

DAC Lock Register

**Figure 15-7. DACLOCK Register**

15	14	13	12	11	10	9	8
KEY				RESERVED			
R-0/W-0h				R-0h			
7	6	5	4	3	2	1	0
RESERVED					DACOUTEN	DACVAL	DACCTL
R-0h					R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 15-10. DACLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	KEY	R-0/W	0h	Writes to this register succeed only if this field is written with a value of 0xA.  Only 16-bit writes will succeed (provided the KEY matches). Read-modify-writes to individual bits in this register will be ignored. Reset type: SYSRSn
11-3	RESERVED	R	0h	Reserved
2	DACOUTEN	R/WOnce	0h	Lock write-access to the DACOUTEN register.  0 DACOUTEN register is not locked. Write 0 to this bit has no effect. 1 DACOUTEN register is locked. Only a system reset can clear this bit. Reset type: SYSRSn
1	DACVAL	R/WOnce	0h	Lock write-access to the DACVALS register.  0 DACVALS register is not locked. Write 0 to this bit has no effect. 1 DACVALS register is locked. Only a system reset can clear this bit. Reset type: SYSRSn
0	DACCTL	R/WOnce	0h	Lock write-access to the DACCTL register.  0 DACCTL register is not locked. Write 0 to this bit has no effect. 1 DACCTL register is locked. Only a system reset can clear this bit. Reset type: SYSRSn

### 15.4.2.7 DACTRIM Register (Offset = 6h) [reset = 0h]

DACTRIM is shown in [Figure 15-8](#) and described in [Table 15-11](#).

Return to the [Summary Table](#).

DAC Trim Register

**Figure 15-8. DACTRIM Register**

15	14	13	12	11	10	9	8
RESERVED				RESERVED			
7	6	5	4	3	2	1	0
OFFSET_TRIM							
R/W-0h							

**Table 15-11. DACTRIM Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-8	RESERVED	R/W	0h	Reserved
7-0	OFFSET_TRIM	R/W	0h	DAC Offset Trim.  This register should not be modified unless specifically indicated by TI Errata or other documentation. Modifying the contents of this register could cause this module to operate outside of datasheet specifications.  Reset type: SYSRSn

### 15.4.3 Register to Driverlib Function Mapping

**Table 15-12. DAC Registers to Driverlib Functions**

File	Driverlib Function
<b>REV</b>	
dac.h	DAC_getRevision
<b>CTL</b>	
dac.h	DAC_setReferenceVoltage
dac.h	DAC_setGainMode
dac.h	DAC_setLoadMode
dac.h	DAC_setPWMSyncSignal
<b>VALA</b>	
dac.h	DAC_getActiveValue
<b>VALS</b>	
dac.h	DAC_setShadowValue
dac.h	DAC_getShadowValue
<b>OUTEN</b>	
dac.h	DAC_enableOutput
dac.h	DAC_disableOutput
<b>LOCK</b>	
dac.h	DAC_lockRegister
dac.h	DAC_isRegisterLocked
<b>TRIM</b>	
dac.c	DAC_tuneOffsetTrim
dac.h	DAC_setOffsetTrim
dac.h	DAC_getOffsetTrim

## Comparator Subsystem (CMPSS)

---



---

The Comparator Subsystem (CMPSS) consists of analog comparators and supporting circuits that are useful for power applications such as peak current mode control, switched-mode power, power factor correction, voltage trip monitoring, and so forth.

Topic	Page
<b>16.1 Introduction</b> .....	<b>1689</b>
<b>16.2 Features</b> .....	<b>1689</b>
<b>16.3 Comparator</b> .....	<b>1690</b>
<b>16.4 Reference DAC</b> .....	<b>1690</b>
<b>16.5 Ramp Generator</b> .....	<b>1692</b>
<b>16.6 Digital Filter</b> .....	<b>1694</b>
<b>16.7 Using the CMPSS</b> .....	<b>1695</b>
<b>16.8 CMPSS Registers</b> .....	<b>1697</b>



## 16.1 Introduction

The comparator subsystem is built around a number of modules. Each subsystem contains two comparators, two reference 12-bit DACs, two digital filters and one ramp generator. Comparators are denoted "H" or "L" within each module where "H" and "L" represent high and low, respectively. Each comparator generates a digital output which indicates whether the voltage on the positive input is greater than the voltage on the negative input. The positive input of the comparator is driven from an external pin or by the PGA (see the *Analog Subsystem* chapter for mux options available to the CMPSS). The negative input can be driven by an external pin or by the programmable reference 12-bit DAC. Each comparator output passes through a programmable digital filter that can remove spurious trip signals. An unfiltered output is also available if filtering is not required. A ramp generator circuit is optionally available to control the reference 12-bit DAC value for the high comparator in the subsystem.

## 16.2 Features

Each CMPSS includes:

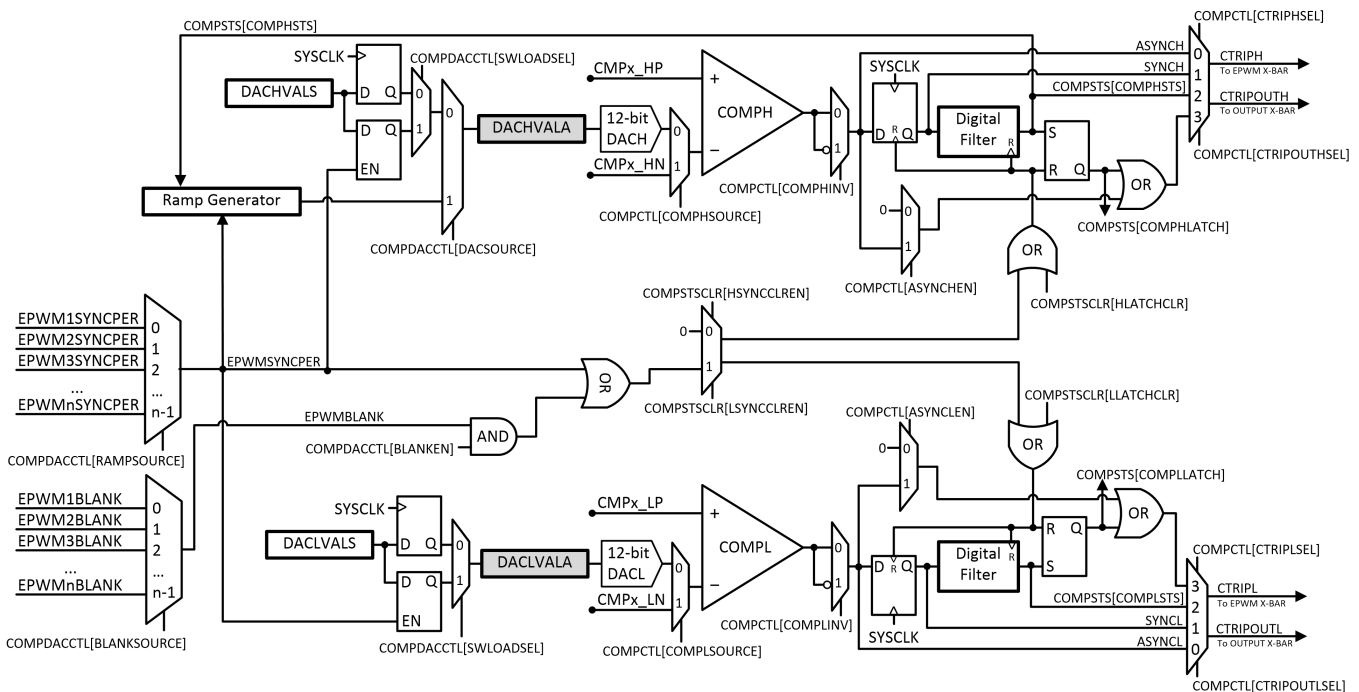
- Two analog comparators
- Two programmable reference 12-bit DACs
- One ramp generator
- Two digital filters
- Ability to synchronize submodules with EPWMSYNCPER
- Ability to extend clear signal with EPWMBLANK
- Ability to synchronize output with SYSCLK
- Ability to latch output
- Ability to invert output
- Option to use hysteresis on the input
- Option for positive input of comparator to be driven by an external signal or by the PGA
- Option for negative input of comparator to be driven by an external signal or by the reference DAC
- Option to choose between VDDA or VDACC to be the DAC reference voltage

### 16.2.1 Block Diagram

The block diagram for the CMPSS is shown in [Figure 16-1](#).

- CTRIPx(x= "H" or "L") signals are connected to the ePWM X-BAR for ePWM trip response. See the *ePWM* chapter for more details on the ePWM X-BAR mux configuration.
- CTRIPxOUTx(x= "H" or "L") signals are connected to the Output X-BAR for external signaling. See the *GPIO* chapter for more details on the Output X-BAR mux configuration.

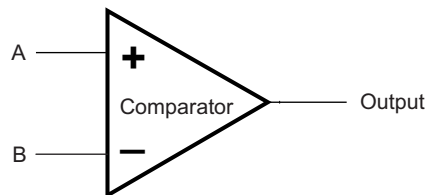
Figure 16-1. CMPSS Module Block Diagram



### 16.3 Comparator

The comparator generates a high digital output when the voltage on the positive input is greater than the voltage on the negative input, and a low digital output when the voltage on the positive input is less than the voltage on the negative input. The comparator is illustrated in Figure 16-2.

Figure 16-2. Comparator Block Diagram



Voltages	Output
Voltage A > Voltage B	1
Voltage A < Voltage B	0

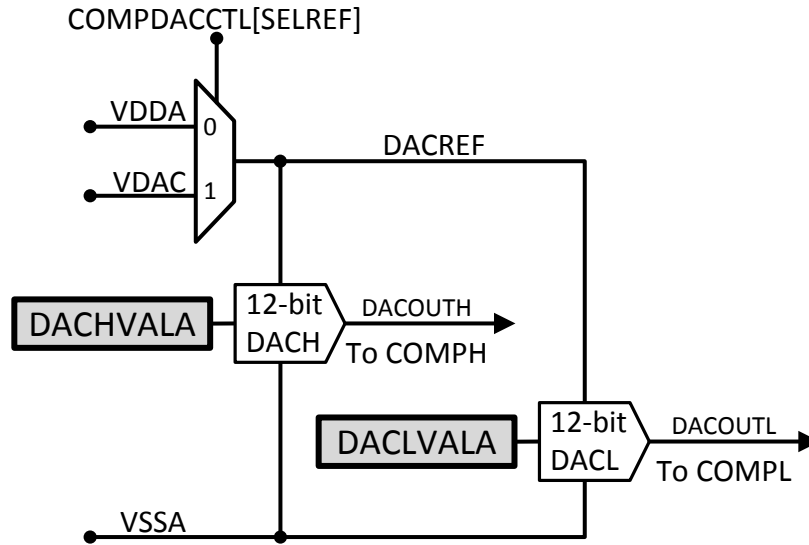
### 16.4 Reference DAC

Each reference 12-bit DAC can be configured to drive a reference voltage into the negative input of its respective comparator. The reference 12-bit DAC output is internal only and cannot be observed externally.

Two sets of DACxVAL registers, DACxVALA and DACxVALS, are present for each reference 12-bit DAC. DACxVALA is a read-only register that actively controls the reference 12-bit DAC value. DACxVALS is a writable shadow register that loads into DACxVALA either immediately or synchronized with the next EPWMSYNCPER event. The high reference 12-bit DAC (DACH) can optionally source its DACHVALA value from the ramp generator instead of DACHVALS.

The operating range of the reference 12-bit DAC is bounded by DACREF and VSSA. The high voltage reference is VDDA by default, but it can be configured to be VDAC. The reference 12-bit DAC is illustrated in [Figure 16-3](#).

**Figure 16-3. Reference DAC Block Diagram**



The ideal output of the reference 12-bit DAC can be calculated as follows:

**Figure 16-4. Output Voltage Calculation**

$$DACOUT = \frac{DACVALA * DACREF}{4096}$$

## 16.5 Ramp Generator

This section discusses the characteristics of the ramp generator and its behavior.

### 16.5.1 Ramp Generator Overview

The ramp generator produces a falling-ramp input for the high reference 12-bit DAC when selected. In this mode, the reference 12-bit DAC uses the most significant 12 bits of the RAMPSTS countdown register as its input. The low 4 bits of the RAMPSTS countdown register effectively act as a prescale for the falling-ramp rate configurable with RAMPDECVALA.

The ramp generator is enabled by setting DACSOURCE = 1. On setting DACSOURCE = 1, the value of RAMPSTS is loaded from RAMPMAXREFS and the register remains static until the selected EPWMSYNCPER signal is received. After receiving the selected EPWMSYNCPER signal, the value of RAMPDECVALA is subtracted from RAMPSTS on every subsequent SYSCLK cycle.

To prevent the subtraction from commencing a SYSCLK cycle after a EPWMSYNCPER event, the RAMPDLYA register which serves as a delay counter can be used to hold off the RAMPSTS subtraction. On receiving a EPWMSYNCPER event, the value of RAMPDLYA is decremented by one on every SYSCLK cycle until the register reaches zero. The RAMPSTS subtraction will only begin when RAMPDLYA is zero.

### 16.5.2 Ramp Generator Behavior

The ramp generator makes state changes on every rising edge of DACSOURCE, EPWMSYNCPER and COMPHSTS.

On the rising edge of DACSOURCE, RAMPMAXREFA, RAMPDECVALA and RAMPDLYA are loaded with their shadow registers. RAMPSTS is loaded with RAMPMAXREFS.

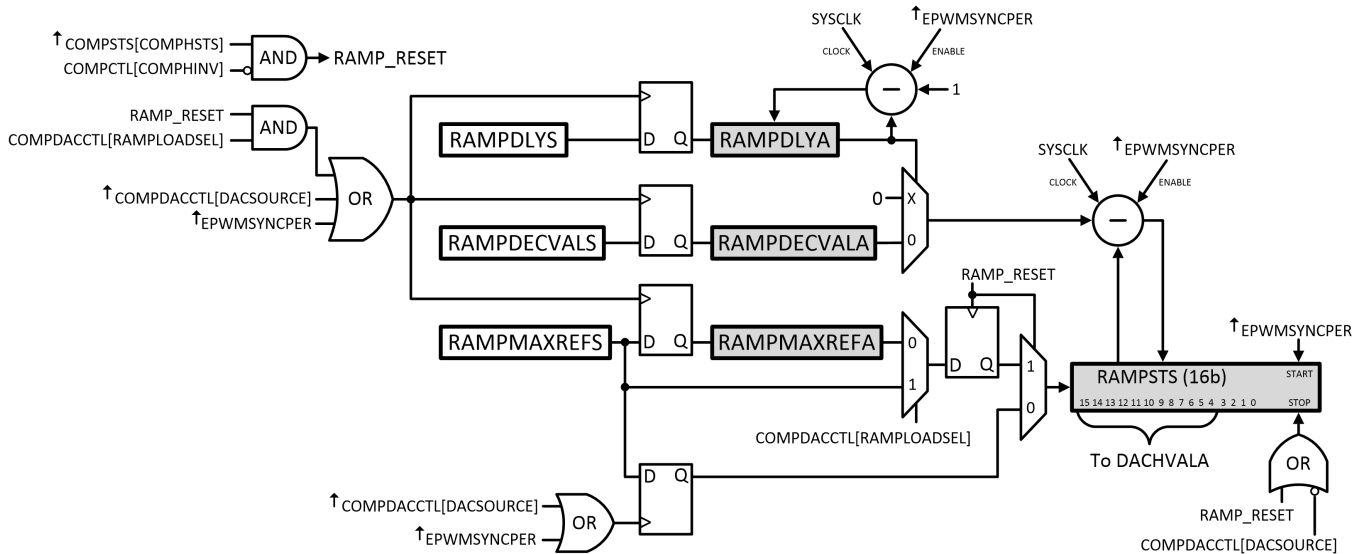
On the rising edge of the selected EPWMSYNCPER, RAMPMAXREFA, RAMPDECVALA and RAMPDLYA are loaded with their shadow registers. RAMPSTS is loaded with RAMPMAXREFS and starts decrementing when RAMPDLYA counter reaches zero.

On the rising edge of COMPHSTS with RAMPLOADSEL = 1, RAMPMAXREFA, RAMPDECVALA and RAMPDLYA are loaded with their shadow registers. RAMPSTS is loaded with RAMPMAXREFS and stops decrementing.

On the rising edge of COMPHSTS with RAMPLOADSEL = 0, RAMPSTS is loaded with RAMPMAXREFA and stops decrementing.

Additionally, if the value of RAMPSTS reaches zero, the RAMPSTS register will remain static at zero until the next EPWMSYNCPER is received. These state changes are illustrated in the ramp generator block diagram in [Figure 16-5](#).

Figure 16-5. Ramp Generator Block Diagram



### 16.5.3 Ramp Generator Behavior at Corner Cases

Since the ramp generator makes state changes on every rising edge of EPWMSYNCPER and COMPHSTS, the following behavior can be expected on instances when these two events occur simultaneously or very close together.

Case 1: COMPHSTS rising edge occurs one or more cycles before EPWMSYNCPER rising edge. RAMPSTS stops decrementing on COMPHSTS rising edge event. RAMPSTS starts decrementing on EPWMSYNCPER rising edge event when RAMPDLYA reaches 0.

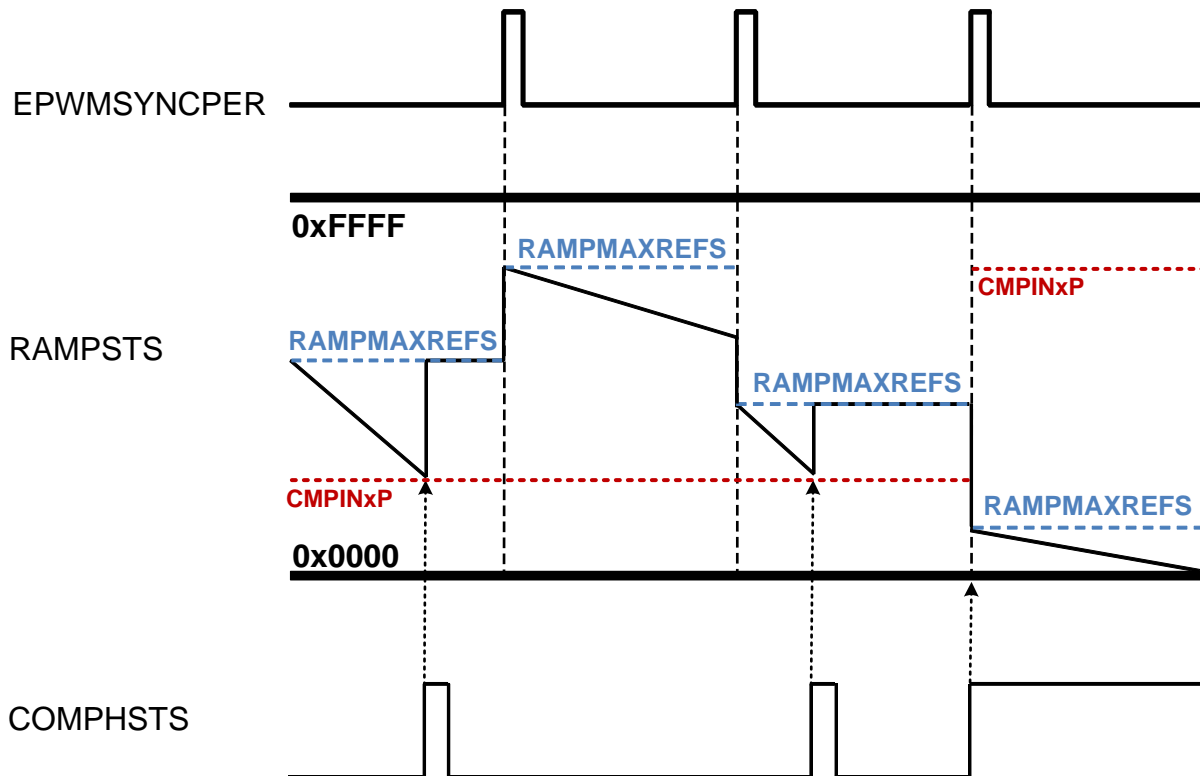
Case 2: COMPHSTS rising edge occurs simultaneously as EPWMSYNCPER rising edge. EPWMSYNCPER rising edge event takes precedence and RAMPSTS starts decrementing when RAMPDLYA reaches 0. COMPHSTS rising edge event is ignored and does not halt RAMPSTS.

Case 3: COMPHSTS rising edge occurs one or more cycles after EPWMSYNCPER rising edge but before RAMPDLYA reaches 0. RAMPSTS does not decrement when RAMPDLYA reaches 0.

Case 4: COMPHSTS rising edge occurs simultaneously as RAMPDLYA reaches 0 from EPWMSYNCPER rising edge. RAMPSTS does not decrement.

This behavior is also illustrated in [Figure 16-6](#).

Figure 16-6. Ramp Generator Behavior



## 16.6 Digital Filter

The digital filter works on a window of FIFO samples (SAMPWIN) taken from the input. The filter output resolves to the majority value of the sample window, where majority is defined by the threshold (THRESH) value. If the majority threshold is not satisfied, the filter output remains unchanged.

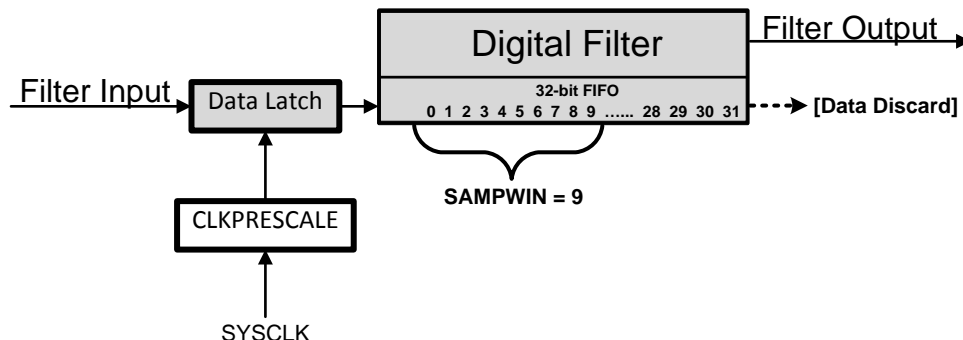
For proper operation, the value of THRESH must be greater than  $SAMPWIN / 2$  and less than or equal to SAMPWIN.

A prescale function (CLKPRESCALE) determines the filter sampling rate, where the filter FIFO captures one sample every prescale system clocks. Old data from the FIFO is discarded.

Note that for SAMPWIN, THRESH and CLKPRESCALE, the internal number used by the digital filter is + 1 in all cases. In essence, samples = SAMPWIN + 1, threshold = THRESH + 1 and prescale = CLKPRESCALE + 1.

A conceptual model of the digital filter is shown in Figure 16-7.

Figure 16-7. Digital Filter Behavior



Equivalent C code of the filter implementation is shown below:

```

if (FILTER_OUTPUT == 0) {
    if (Num_1s_in_SAMPWIN >= THRESH) {
        FILTER_OUTPUT = 1;
    }
}
else {
    if (Num_0s_in_SAMPWIN >= THRESH) {
        FILTER_OUTPUT = 0;
    }
}

```

### 16.6.1 Filter Initialization Sequence

To ensure proper operation of the digital filter, the following initialization sequence is recommended:

1. Configure and enable the comparator for operation
2. Configure the digital filter parameters for operation
  - Set SAMPWIN for the number of samples to monitor in the FIFO window
  - Set THRESH for the threshold required for majority qualification
  - Set CLKPRESCALE for the digital filter clock prescale value
3. Initialize the sample values in the digital FIFO window by setting FILINIT
4. Clear COMPSTS latch via COMPSTSCLR if the latched path is desired
5. Configure the CTRIP and CTRIPOUT signal paths
6. If desired, configure the ePWM and GPIO modules to accept the filtered signals

## 16.7 Using the CMPSS

### 16.7.1 LATCHCLR, EPWMSYNCPER and EPWMBLANK Signals

The LATCHCLR signal initializes the digital filter to the output of the synchronization block and holds the latch output in reset (0) with the output of the synchronization block reflecting the async output of the comparator. It is activated in software using xLATCHCLR(x= "H" or "L"). It can also be activated by EPWMSYNCPER when xSYNCCLEN(x= "H" or "L") is set. If a longer LATCHCLR signal is required, the EPWMBLANK signal can be used to extend it by setting BLANKEN.

EPWMSYNCPER and EPWMBLANK (BLANKWDW) come from the Time-Base and Digital Compare submodules of the EPWM respectively. For a detailed description of how these two signals are generated, refer to the respective submodule chapter of the EPWM.

The EPWMSYNCPER signal that loads DACxVALA when COMPDACCTL [SWLOADSEL] = 1 is a level trigger load. If TBCTR and TBPRD of the EPWM are both 0, EPWMSYNCPER will be held at level high and DACxVALA will be loaded immediately from DACxVALS irrespective of the value of COMPDACCTL [SWLOADSEL]. Due to this, it is recommended to configure the EPWM first before setting COMPDACCTL [SWLOADSEL] to 1.

---

**NOTE:** The name of the sync signal that the CMPSS receives from the EPWM has been updated from PWMSYNC to EPWMSYNCPER (SYNCPER/PWMSYNCPER/EPWMxSYNCPER) to avoid confusion with the other EPWM sync signals EPWMSYNCI and EPWMSYNCO. For a description of what these signals are, see the EPWM chapter.

---

### 16.7.2 Synchronizer, Digital Filter and Latch Delays

The synchronization block adds a delay of 1-2 sysclks. If the digital filter is bypassed (all filter settings are 0), the digital filter will add a delay of 2 sysclks. The latch adds 1 sysclk delay.

### 16.7.3 Calibrating the CMPSS

The CMPSS has two sources of offset errors: comparator offset error and compdac offset error. In the data manual, the comparator offset error is referred to as **Input referred offset error** and compdac offset error is referred to as **Static offset error**. See the device specific data manual for their values.

If both inputs of the comparator are driven from a pin, only the comparator offset error applies. However if the inverting input of the comparator is driven from the compdac, then only the compdac offset error applies. This is because the compdac offset error includes comparator offset error.

Due to the offset errors, it is recommended that the CMPSS be calibrated to ensure trips happen at the expected levels. The flow below outlines how the calibration can be performed if the inverting input of the comparator is driven from the compdac.

Notes before calibration:

1. A static DC signal is required on the non-inverting input of the comparator.
2. Hysteresis should be disabled for calibration. It can be re-enabled after calibration is complete.
3. A noisy input can make calibration difficult so it is recommended to use the latch with non-zero filter settings depending on how noisy the signal on the non-inverting input is.

This approach sweeps down the compdac:

1. Set the starting compdac value to max, 0xFFFF.
  - Optional: Instead of setting the starting compdac value to max, it can be set to **Vtarget + Static offset error + Margin**. Where **Vtarget** is the approximate DC voltage on the non-inverting input, **Static offset error** is the compdac offset error specification and **Margin** is some amount of guard band. This can lead to a faster calibration but will only work if **Vtarget** is known. Alternatively if **Vtarget** is unknown, the ADC can be used to convert it.
2. Decrement compdac value by 1.
3. Wait for compdac to settle.
4. Clear latch.
5. Wait for possible latch set.
6. If latch is set, trip code is found exit.
  - Optional: The trip code can be double checked by:
    - I. Increasing compdac value by 1.
    - II. Clear latch.
    - III. Wait for possible latch set.
    - IV. Latch should be unset.
7. If latch is unset, go back to step 2 and repeat.

It is also possible to calibrate the CMPSS if both inputs of the comparator are driven from a pin. For this case, the flow stays the same but the voltage on the inverting pin of the comparator is swept externally.

### 16.7.4 Enabling and Disabling the CMPSS Clock

If the clock to the CMPSS module is disabled while the comparator is active, the following behavior can be expected:

- The comparator remains unaffected and will continue to trip from voltages on its inputs.
- If the reference 12-bit DAC is driving the negative input of the comparator, the voltage on the negative input remains static and unaffected but DACVALA would no longer be updated from the ramp generator or DACVALS.
- The ramp generator, synchronize block and digital filter freeze on their current states.

Enabling the clock to the CMPSS restores it to the state before the clock was disabled.



## 16.8 CMPSS Registers

This section describes the CMPSS Registers.

### 16.8.1 CMPSS Base Addresses

**Table 16-1. CMPSS Base Address Table**

Device Registers	Register Name	Start Address	End Address
Cmpss1Regs	CMPSS_REGS	0x0000_5C80	0x0000_5C9F
Cmpss2Regs	CMPSS_REGS	0x0000_5CA0	0x0000_5CBF
Cmpss3Regs	CMPSS_REGS	0x0000_5CC0	0x0000_5CDF
Cmpss4Regs	CMPSS_REGS	0x0000_5CE0	0x0000_5CFF
Cmpss5Regs	CMPSS_REGS	0x0000_5D00	0x0000_5D1F
Cmpss6Regs	CMPSS_REGS	0x0000_5D20	0x0000_5D3F
Cmpss7Regs	CMPSS_REGS	0x0000_5D40	0x0000_5D5F

## 16.8.2 CMPSS\_REGS Registers

Table 16-2 lists the CMPSS\_REGS registers. All register offset addresses not listed in Table 16-2 should be considered as reserved locations and the register contents should not be modified.

**Table 16-2. CMPSS\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	COMPCTL	CMPSS Comparator Control Register	EALLOW	<a href="#">Go</a>
1h	COMPHYSCTL	CMPSS Comparator Hysteresis Control Register	EALLOW	<a href="#">Go</a>
2h	COMPSTS	CMPSS Comparator Status Register		<a href="#">Go</a>
3h	COMPSTCLR	CMPSS Comparator Status Clear Register	EALLOW	<a href="#">Go</a>
4h	COMPDACCTL	CMPSS DAC Control Register	EALLOW	<a href="#">Go</a>
6h	DACHVALS	CMPSS High DAC Value Shadow Register		<a href="#">Go</a>
7h	DACHVALA	CMPSS High DAC Value Active Register		<a href="#">Go</a>
8h	RAMPMAXREFA	CMPSS Ramp Max Reference Active Register		<a href="#">Go</a>
Ah	RAMPMAXREFS	CMPSS Ramp Max Reference Shadow Register		<a href="#">Go</a>
Ch	RAMPDECVALA	CMPSS Ramp Decrement Value Active Register		<a href="#">Go</a>
Eh	RAMPDECVALS	CMPSS Ramp Decrement Value Shadow Register		<a href="#">Go</a>
10h	RAMPSTS	CMPSS Ramp Status Register		<a href="#">Go</a>
12h	DACLVALS	CMPSS Low DAC Value Shadow Register		<a href="#">Go</a>
13h	DACLVALA	CMPSS Low DAC Value Active Register		<a href="#">Go</a>
14h	RAMPDLYA	CMPSS Ramp Delay Active Register		<a href="#">Go</a>
15h	RAMPDLYS	CMPSS Ramp Delay Shadow Register		<a href="#">Go</a>
16h	CTRIPLFILCTL	CTRIPL Filter Control Register	EALLOW	<a href="#">Go</a>
17h	CTRIPLFILCLKCTL	CTRIPL Filter Clock Control Register	EALLOW	<a href="#">Go</a>
18h	CTRIPHFILCTL	CTRIPH Filter Control Register	EALLOW	<a href="#">Go</a>
19h	CTRIPHFILCLKCTL	CTRIPH Filter Clock Control Register	EALLOW	<a href="#">Go</a>
1Ah	COMPLOCK	CMPSS Lock Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 16-3 shows the codes that are used for access types in this section.

**Table 16-3. CMPSS\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
WOnce	WOnce	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		

**Table 16-3. CMPSS\_REGS Access Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**16.8.2.1 COMPCTL Register (Offset = 0h) [reset = 0h]**

 COMPCTL is shown in [Figure 16-8](#) and described in [Table 16-4](#).

 Return to the [Summary Table](#).

CMPSS Comparator Control Register

**Figure 16-8. COMPCTL Register**

15	14	13	12	11	10	9	8
COMPDA CE	ASYNCL EN	CTRIPOUTLSEL		CTRIPLSEL		COMPLIN V	COMPLSOUR CE
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
RESERVED	ASYNCH EN	CTRIPOUTHSEL		CTRI PHSEL		COMPHIN V	COMPHSOUR CE
R-0h	R/W-0h	R/W-0h		R/W-0h		R/W-0h	R/W-0h

**Table 16-4. COMPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	COMPDA CE	R/W	0h	Comparator/DAC enable.  0 Comparator/DAC disabled 1 Comparator/DAC enabled Reset type: SYSRSn
14	ASYNCL EN	R/W	0h	Low comparator asynchronous path enable. Allows asynchronous comparator output to feed into OR gate with latched digital filter signal when CTRIPLSEL=3 or CTRIPOUTLSEL=3.  0 Asynchronous comparator output does not feed into OR gate with latched digital filter output 1 Asynchronous comparator output feeds into OR gate with latched digital filter output Reset type: SYSRSn
13-12	CTRIPOUTLSEL	R/W	0h	Low comparator CTRIPOUTL source select.  0 Asynchronous comparator output drives CTRIPOUTL 1 Synchronous comparator output drives CTRIPOUTL 2 Output of digital filter drives CTRIPOUTL 3 Latched output of digital filter drives CTRIPOUTL Reset type: SYSRSn
11-10	CTRIPLSEL	R/W	0h	Low comparator CTRIPL source select.  0 Asynchronous comparator output drives CTRIPL 1 Synchronous comparator output drives CTRIPL 2 Output of digital filter drives CTRIPL 3 Latched output of digital filter drives CTRIPL Reset type: SYSRSn
9	COMPLIN V	R/W	0h	Low comparator output invert.  0 Output of comparator is not inverted 1 Output of comparator is inverted Reset type: SYSRSn
8	COMPLSOUR CE	R/W	0h	Low comparator input source.  0 Inverting input of comparator driven by internal DAC 1 Inverting input of comparator driven through external pin Reset type: SYSRSn
7	RESERVED	R	0h	Reserved

**Table 16-4. COMPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	ASYNCHEN	R/W	0h	<p>High comparator asynchronous path enable. Allows asynchronous comparator output to feed into OR gate with latched digital filter signal when CTRIPHSEL=3 or CTRIPOUTHSEL=3.</p> <p>0 Asynchronous comparator output does not feed into OR gate with latched digital filter output            1 Asynchronous comparator output feeds into OR gate with latched digital filter output            Reset type: SYSRSn</p>
5-4	CTRIPOUTHSEL	R/W	0h	<p>High comparator CTRIPOUTH source select.</p> <p>0 Asynchronous comparator output drives CTRIPOUTH            1 Synchronous comparator output drives CTRIPOUTH            2 Output of digital filter drives CTRIPOUTH            3 Latched output of digital filter drives CTRIPOUTH            Reset type: SYSRSn</p>
3-2	CTRIPHSEL	R/W	0h	<p>High comparator CTRIPH source select.</p> <p>0 Asynchronous comparator output drives CTRIPH            1 Synchronous comparator output drives CTRIPH            2 Output of digital filter drives CTRIPH            3 Latched output of digital filter drives CTRIPH            Reset type: SYSRSn</p>
1	COMPHINV	R/W	0h	<p>High comparator output invert.</p> <p>0 Output of comparator is not inverted            1 Output of comparator is inverted            Reset type: SYSRSn</p>
0	COMPHSOURCE	R/W	0h	<p>High comparator input source.</p> <p>0 Inverting input of comparator driven by internal DAC            1 Inverting input of comparator driven through external pin            Reset type: SYSRSn</p>

**16.8.2.2 COMPHYCTL Register (Offset = 1h) [reset = 0h]**

COMPHYCTL is shown in [Figure 16-9](#) and described in [Table 16-5](#).

Return to the [Summary Table](#).

CMPSS Comparator Hysteresis Control Register

**Figure 16-9. COMPHYCTL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					COMPHYCTL		
R-0h					R/W-0h		

**Table 16-5. COMPHYCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2-0	COMPHYCTL	R/W	0h	Comparator hysteresis. Sets the amount of hysteresis on the comparator inputs.  0 None 1 Set to typical hysteresis 2 Set to 2x of typical hysteresis 3 Set to 3x of typical hysteresis 4 Set to 4x of typical hysteresis Reset type: SYSRStn

### 16.8.2.3 COMPSTS Register (Offset = 2h) [reset = 0h]

COMPSTS is shown in [Figure 16-10](#) and described in [Table 16-6](#).

Return to the [Summary Table](#).

CMPSS Comparator Status Register

**Figure 16-10. COMPSTS Register**

15	14	13	12	11	10	9	8
RESERVED						COMPLLATCH	COMPLSTS
R-0h						R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED						COMPHLATCH	COMPSTS
R-0h						R-0h	R-0h

**Table 16-6. COMPSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	COMPLLATCH	R	0h	Latched value of low comparator digital filter output Reset type: SYSRSn
8	COMPLSTS	R	0h	Low comparator digital filter output Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	COMPHLATCH	R	0h	Latched value of high comparator digital filter output Reset type: SYSRSn
0	COMPSTS	R	0h	High comparator digital filter output Reset type: SYSRSn

**16.8.2.4 COMPSTSCLR Register (Offset = 3h) [reset = 0h]**

 COMPSTSCLR is shown in [Figure 16-11](#) and described in [Table 16-7](#).

 Return to the [Summary Table](#).

CMPSS Comparator Status Clear Register

**Figure 16-11. COMPSTSCLR Register**

15	14	13	12	11	10	9	8
RESERVED					LSYNCCLREN	LLATCHCLR	RESERVED
R-0h					R/W-0h	R-0/W1S-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED					HSYNCCLREN	HLATCHCLR	RESERVED
R-0h					R/W-0h	R-0/W1S-0h	R-0h

**Table 16-7. COMPSTSCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved
10	LSYNCCLREN	R/W	0h	Low comparator latch EPWMSYNCPER clear. Enable EPWMSYNCPER reset of low comparator digital filter output latch COMPSTS[COMPLLATCH].  0 EPWMSYNCPER will not reset latch 1 EPWMSYNCPER will reset latch Reset type: SYSRSn
9	LLATCHCLR	R-0/W1S	0h	Low comparator latch software clear. Perform software reset of low comparator digital filter output latch COMPSTS[COMPLLATCH]. Reads always return 0.  0 No effect 1 Generate a single pulse of latch reset signal for COMPSTS[COMPLLATCH] Reset type: SYSRSn
8-3	RESERVED	R	0h	Reserved
2	HSYNCCLREN	R/W	0h	High comparator latch EPWMSYNCPER clear. Enable EPWMSYNCPER reset of high comparator digital filter output latch COMPSTS[COMPHLATCH].  0 EPWMSYNCPER will not reset latch 1 EPWMSYNCPER will reset latch Reset type: SYSRSn
1	HLATCHCLR	R-0/W1S	0h	High comparator latch software clear. Perform software reset of high comparator digital filter output latch COMPSTS[COMPHLATCH]. Reads always return 0.  0 No effect 1 Generate a single pulse of latch reset signal for COMPSTS[COMPHLATCH] Reset type: SYSRSn
0	RESERVED	R	0h	Reserved



### 16.8.2.5 COMPDACCTL Register (Offset = 4h) [reset = 0h]

COMPDACCTL is shown in [Figure 16-12](#) and described in [Table 16-8](#).

Return to the [Summary Table](#).

CMPSS DAC Control Register

**Figure 16-12. COMPDACCTL Register**

15		14		13		12		11		10		9		8	
FREESOFT				RESERVED		BLANKEN		BLANKSOURCE							
R/W-0h				R-0h		R/W-0h		R/W-0h							
7		6		5		4		3		2		1		0	
SWLOADSEL		RAMPLOADSEL		SELREF		RAMPSOURCE						DACSOURCE			
R/W-0h		R/W-0h		R/W-0h		R/W-0h						R/W-0h			

**Table 16-8. COMPDACCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	FREESOFT	R/W	0h	Free-run or software-run emulation behavior. Behavior of the ramp generator during emulation suspend.  00b Ramp generator stops immediately during emulation suspend 01b Ramp generator completes current ramp and stops at next EPWMSYNCPER during emulation suspend 1Xb Ramp generator runs freely Reset type: SYSRSn
13	RESERVED	R	0h	Reserved
12	BLANKEN	R/W	0h	EPWMBLANK enable. This bit enables the EPWMBLANK signal.  0 EPWMBLANK signal is disabled. 1 EPWMBLANK signal is enabled. Reset type: SYSRSn
11-8	BLANKSOURCE	R/W	0h	EPWMBLANK source select. This bit field determines which EPWMnBLANK is passed on as the EPWMBLANK signal.  Where n represents the maximum number of EPWMBLANK signals available on the device:  0 EPWM1BLANK 1 EPWM2BLANK 2 EPWM3BLANK ... n-1 EPWMnBLANK Reset type: SYSRSn
7	SWLOADSEL	R/W	0h	Software load select. Determines whether DACxVALA is updated from DACxVALS on SYSCLK or EPWMSYNCPER.  0 DACxVALA is updated from DACxVALS on SYSCLK 1 DACxVALA is updated from DACxVALS on EPWMSYNCPER Reset type: SYSRSn
6	RAMPLOADSEL	R/W	0h	Ramp load select. Determines whether RAMPSTS is updated from RAMPMAXREFA or RAMPMAXREFS when COMPSTS[COMPSTS] is triggered.  0 RAMPSTS is loaded from RAMPMAXREFA 1 RAMPSTS is loaded from RAMPMAXREFS Reset type: SYSRSn

**Table 16-8. COMPDACCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	SELREF	R/W	0h	DAC reference select. Determines which voltage supply is used as the reference for the internal comparator DACs.  0 VDDA is the voltage reference for the DAC 1 VDAC is the voltage reference for the DAC Reset type: SYSRSn
4-1	RAMPSOURCE	R/W	0h	Ramp generator source select. Determines which EPWMSYNCPER signal is used within the CMPSS module.  Where n represents the maximum number of EPWMSYNCPER signals available on the device:  0 EPWM1SYNCPER 1 EPWM2SYNCPER 2 EPWM3SYNCPER ... n-1 EPWMnSYNCPER Reset type: SYSRSn
0	DACSOURCE	R/W	0h	DAC source select. Determines whether DACHVALA is updated from DACHVALS or from the ramp generator.  0 DACHVALA is updated from DACHVALS 1 DACHVALA is updated from the ramp generator Reset type: SYSRSn

### 16.8.2.6 DACHVALS Register (Offset = 6h) [reset = 0h]

DACHVALS is shown in [Figure 16-13](#) and described in [Table 16-9](#).

Return to the [Summary Table](#).

CMPSS High DAC Value Shadow Register

**Figure 16-13. DACHVALS Register**

15	14	13	12	11	10	9	8
RESERVED				DACVAL			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
DACVAL							
R/W-0h							

**Table 16-9. DACHVALS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R/W	0h	High DAC shadow value. When COMPDACCTL[DACSOURCE]=0, the value of DACHVALS is loaded into DACHVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL]. Reset type: SYSRSn

### 16.8.2.7 DACHVALA Register (Offset = 7h) [reset = 0h]

DACHVALA is shown in [Figure 16-14](#) and described in [Table 16-10](#).

Return to the [Summary Table](#).

CMPSS High DAC Value Active Register

**Figure 16-14. DACHVALA Register**

15	14	13	12	11	10	9	8
RESERVED				DACVAL			
R-0h				R-0h			
7	6	5	4	3	2	1	0
DACVAL							
R-0h							

**Table 16-10. DACHVALA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R	0h	High DAC active value. Value that is actively driven by the high DAC. Reset type: SYSRSn

**16.8.2.8 RAMPMAXREFA Register (Offset = 8h) [reset = 0h]**

RAMPMAXREFA is shown in [Figure 16-15](#) and described in [Table 16-11](#).

Return to the [Summary Table](#).

CMPSS Ramp Max Reference Active Register

**Figure 16-15. RAMPMAXREFA Register**

15	14	13	12	11	10	9	8
RAMPMAXREF							
R-0h							
7	6	5	4	3	2	1	0
RAMPMAXREF							
R-0h							

**Table 16-11. RAMPMAXREFA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPMAXREF	R	0h	Ramp maximum reference active value. Latched value to be loaded into ramp generator RAMPSTS. Reset type: SYSRSn

**16.8.2.9 RAMPMAXREFS Register (Offset = Ah) [reset = 0h]**

RAMPMAXREFS is shown in [Figure 16-16](#) and described in [Table 16-12](#).

Return to the [Summary Table](#).

CMPSS Ramp Max Reference Shadow Register

**Figure 16-16. RAMPMAXREFS Register**

15	14	13	12	11	10	9	8
RAMPMAXREF							
R/W-0h							
7	6	5	4	3	2	1	0
RAMPMAXREF							
R/W-0h							

**Table 16-12. RAMPMAXREFS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPMAXREF	R/W	0h	Ramp maximum reference shadow. Unlatched value to be loaded into ramp generator RAMPSTS. Reset type: SYSRSn

**16.8.2.10 RAMPDECVALA Register (Offset = Ch) [reset = 0h]**

RAMPDECVALA is shown in [Figure 16-17](#) and described in [Table 16-13](#).

Return to the [Summary Table](#).

CMPSS Ramp Decrement Value Active Register

**Figure 16-17. RAMPDECVALA Register**

15	14	13	12	11	10	9	8
RAMPDECVAL							
R-0h							
7	6	5	4	3	2	1	0
RAMPDECVAL							
R-0h							

**Table 16-13. RAMPDECVALA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPDECVAL	R	0h	Ramp decrement value active. Latched value that will be subtracted from RAMPSTS. Reset type: SYSRSn

**16.8.2.11 RAMPDECVALS Register (Offset = Eh) [reset = 0h]**

RAMPDECVALS is shown in [Figure 16-18](#) and described in [Table 16-14](#).

Return to the [Summary Table](#).

CMPSS Ramp Decrement Value Shadow Register

**Figure 16-18. RAMPDECVALS Register**

15	14	13	12	11	10	9	8
RAMPDECVAL							
R/W-0h							
7	6	5	4	3	2	1	0
RAMPDECVAL							
R/W-0h							

**Table 16-14. RAMPDECVALS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPDECVAL	R/W	0h	Ramp decrement value shadow. Unlatched value to be loaded into RAMPDECVALA. Reset type: SYSRSn



**16.8.2.12 RAMPSTS Register (Offset = 10h) [reset = 0h]**

RAMPSTS is shown in [Figure 16-19](#) and described in [Table 16-15](#).

Return to the [Summary Table](#).

CMPSS Ramp Status Register

**Figure 16-19. RAMPSTS Register**

15	14	13	12	11	10	9	8
RAMPVALUE							
R-0h							
7	6	5	4	3	2	1	0
RAMPVALUE							
R-0h							

**Table 16-15. RAMPSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPVALUE	R	0h	Ramp value. Present value of ramp generator. Reset type: SYSRSn

**16.8.2.13 DACLVALS Register (Offset = 12h) [reset = 0h]**

DACLVALS is shown in [Figure 16-20](#) and described in [Table 16-16](#).

Return to the [Summary Table](#).

CMPSS Low DAC Value Shadow Register

**Figure 16-20. DACLVALS Register**

15	14	13	12	11	10	9	8
RESERVED				DACVAL			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
DACVAL							
R/W-0h							

**Table 16-16. DACLVALS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R/W	0h	Low DAC shadow value. value to be loaded into DACLVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL]. Reset type: SYSRSn

**16.8.2.14 DACLVALA Register (Offset = 13h) [reset = 0h]**

DACLVALA is shown in [Figure 16-21](#) and described in [Table 16-17](#).

Return to the [Summary Table](#).

CMPSS Low DAC Value Active Register

**Figure 16-21. DACLVALA Register**

15	14	13	12	11	10	9	8
RESERVED				DACVAL			
R-0h				R-0h			
7	6	5	4	3	2	1	0
DACVAL							
R-0h							

**Table 16-17. DACLVALA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R	0h	Low DAC active value. Value that is actively driven by the low DAC. Reset type: SYSRSn

**16.8.2.15 RAMPDLYA Register (Offset = 14h) [reset = 0h]**

RAMPDLYA is shown in [Figure 16-22](#) and described in [Table 16-18](#).

Return to the [Summary Table](#).

CMPSS Ramp Delay Active Register

**Figure 16-22. RAMPDLYA Register**

15	14	13	12	11	10	9	8
RESERVED			DELAY				
R-0h			R-0h				
7	6	5	4	3	2	1	0
DELAY							
R-0h							

**Table 16-18. RAMPDLYA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-0	DELAY	R	0h	Ramp delay active value. Latched value of the number of cycles to delay the start of the ramp generator decremter after a EPWMSYNCPER is received. Reset type: SYSRSn

**16.8.2.16 RAMPDLYS Register (Offset = 15h) [reset = 0h]**

RAMPDLYS is shown in [Figure 16-23](#) and described in [Table 16-19](#).

Return to the [Summary Table](#).

CMPSS Ramp Delay Shadow Register

**Figure 16-23. RAMPDLYS Register**

15	14	13	12	11	10	9	8
RESERVED			DELAY				
R-0h			R/W-0h				
7	6	5	4	3	2	1	0
DELAY							
R/W-0h							

**Table 16-19. RAMPDLYS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-0	DELAY	R/W	0h	Ramp delay shadow value. Unlatched value to be loaded into RAMPDLYA. Reset type: SYSRSn

**16.8.2.17 CTRIPLFILCTL Register (Offset = 16h) [reset = 0h]**

CTRIPLFILCTL is shown in [Figure 16-24](#) and described in [Table 16-20](#).

Return to the [Summary Table](#).

CTRIPL Filter Control Register

**Figure 16-24. CTRIPLFILCTL Register**

15	14	13	12	11	10	9	8
FILINIT	RESERVED	THRESH				SAMPWIN	
R-0/W1S-0h	R-0h	R/W-0h				R/W-0h	
7	6	5	4	3	2	1	0
SAMPWIN				RESERVED			
R/W-0h				R-0h			

**Table 16-20. CTRIPLFILCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	FILINIT	R-0/W1S	0h	Low filter initialization.  0 No effect 1 Initialize all samples to the filter input value Reset type: SYSRSn
14	RESERVED	R	0h	Reserved
13-9	THRESH	R/W	0h	Low filter majority voting threshold. At least THRESH samples of the opposite state must appear within the sample window in order for the output to change state. Threshold used is THRESH+1. Reset type: SYSRSn
8-4	SAMPWIN	R/W	0h	Low filter sample window size. Number of samples to monitor is SAMPWIN+1. Reset type: SYSRSn
3-0	RESERVED	R	0h	Reserved

**16.8.2.18 CTRIPLFILCLKCTL Register (Offset = 17h) [reset = 0h]**

CTRIPLFILCLKCTL is shown in [Figure 16-25](#) and described in [Table 16-21](#).

Return to the [Summary Table](#).

CTRIPL Filter Clock Control Register

**Figure 16-25. CTRIPLFILCLKCTL Register**

15	14	13	12	11	10	9	8
RESERVED						CLKPRESCALE	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
CLKPRESCALE							
R/W-0h							

**Table 16-21. CTRIPLFILCLKCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	CLKPRESCALE	R/W	0h	Low filter sample clock prescale. Number of system clocks between samples is CLKPRESCALE+1. Reset type: SYSRSn

**16.8.2.19 CTRIPFILCTL Register (Offset = 18h) [reset = 0h]**

CTRIPFILCTL is shown in [Figure 16-26](#) and described in [Table 16-22](#).

Return to the [Summary Table](#).

CTRIPH Filter Control Register

**Figure 16-26. CTRIPFILCTL Register**

15	14	13	12	11	10	9	8
FILINIT	RESERVED	THRESH				SAMPWIN	
R-0/W1S-0h	R-0h	R/W-0h				R/W-0h	
7	6	5	4	3	2	1	0
SAMPWIN				RESERVED			
R/W-0h				R-0h			

**Table 16-22. CTRIPFILCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	FILINIT	R-0/W1S	0h	High filter initialization.  0 No effect 1 Initialize all samples to the filter input value Reset type: SYSRSn
14	RESERVED	R	0h	Reserved
13-9	THRESH	R/W	0h	High filter majority voting threshold. At least THRESH samples of the opposite state must appear within the sample window in order for the output to change state. Threshold used is THRESH+1. Reset type: SYSRSn
8-4	SAMPWIN	R/W	0h	High filter sample window size. Number of samples to monitor is SAMPWIN+1. Reset type: SYSRSn
3-0	RESERVED	R	0h	Reserved



**16.8.2.20 CTRIPFILCLKCTL Register (Offset = 19h) [reset = 0h]**

CTRIPFILCLKCTL is shown in [Figure 16-27](#) and described in [Table 16-23](#).

Return to the [Summary Table](#).

CTRIPH Filter Clock Control Register

**Figure 16-27. CTRIPFILCLKCTL Register**

15	14	13	12	11	10	9	8
RESERVED						CLKPRESCALE	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
CLKPRESCALE							
R/W-0h							

**Table 16-23. CTRIPFILCLKCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	CLKPRESCALE	R/W	0h	High filter sample clock prescale. Number of system clocks between samples is CLKPRESCALE+1. Reset type: SYSRSn

**16.8.2.21 COMPLOCK Register (Offset = 1Ah) [reset = 0h]**

 COMPLOCK is shown in [Figure 16-28](#) and described in [Table 16-24](#).

 Return to the [Summary Table](#).

CMPSS Lock Register

**Figure 16-28. COMPLOCK Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			RESERVED	CTRIIP	DACCTL	COMPHYSCTL	COMPCTL
R-0h				R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 16-24. COMPLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R/WOnce	0h	Reserved
3	CTRIIP	R/WOnce	0h	Lock write-access to the CTRIPxFILTCTL and CTRIPxFILCLKCTL registers.  0 CTRIPxFILTCTL and CTRIPxFILCLKCTL registers are not locked. Write 0 to this bit has no effect. 1 CTRIPxFILTCTL and CTRIPxFILCLKCTL registers are locked. Only a system reset can clear this bit. Reset type: SYSRSn
2	DACCTL	R/WOnce	0h	Lock write-access to the DACCTL register.  0 DACCTL register is not locked. Write 0 to this bit has no effect. 1 DACCTL register is locked. Only a system reset can clear this bit. Reset type: SYSRSn
1	COMPHYSCTL	R/WOnce	0h	Lock write-access to the COMPHYSCTL register.  0 COMPHYSCTL register is not locked. Write 0 to this bit has no effect. 1 COMPHYSCTL register is locked. Only a system reset can clear this bit. Reset type: SYSRSn
0	COMPCTL	R/WOnce	0h	Lock write-access to the COMPCTL register.  0 COMPCTL register is not locked. Write 0 to this bit has no effect. 1 COMPCTL register is locked. Only a system reset can clear this bit. Reset type: SYSRSn

### 16.8.3 Register to Driverlib Function Mapping

**Table 16-25. CMPSS Registers to Driverlib Functions**

File	Driverlib Function
<b>COMPCTL</b>	
cmpss.h	CMPSS_enableModule
cmpss.h	CMPSS_disableModule
cmpss.h	CMPSS_configHighComparator
cmpss.h	CMPSS_configLowComparator
cmpss.h	CMPSS_configOutputsHigh
cmpss.h	CMPSS_configOutputsLow
<b>COMPHYSCTL</b>	
cmpss.h	CMPSS_setHysteresis
<b>COMPSTS</b>	
cmpss.c	CMPSS_configLatchOnPWMSYNC
cmpss.h	CMPSS_getStatus
cmpss.h	CMPSS_clearFilterLatchHigh
cmpss.h	CMPSS_clearFilterLatchLow
<b>COMPSTSLR</b>	
cmpss.c	CMPSS_configLatchOnPWMSYNC
cmpss.h	CMPSS_clearFilterLatchHigh
cmpss.h	CMPSS_clearFilterLatchLow
<b>COMPDACCTL</b>	
cmpss.c	CMPSS_configRamp
cmpss.h	CMPSS_configDAC
cmpss.h	CMPSS_configBlanking
cmpss.h	CMPSS_enableBlanking
cmpss.h	CMPSS_disableBlanking
<b>DACHVALS</b>	
cmpss.h	CMPSS_setDACValueHigh
<b>DACHVALA</b>	
cmpss.h	CMPSS_getDACValueHigh
<b>RAMPMAXREFA</b>	
cmpss.h	CMPSS_getMaxRampValue
<b>RAMPMAXREFS</b>	
cmpss.c	CMPSS_configRamp
cmpss.h	CMPSS_setMaxRampValue
<b>RAMPDECVALA</b>	
cmpss.h	CMPSS_getRampDecValue
<b>RAMPDECVALS</b>	
cmpss.c	CMPSS_configRamp
cmpss.h	CMPSS_setRampDecValue
<b>DACLVALS</b>	
cmpss.h	CMPSS_setDACValueLow
<b>DACLVALA</b>	
cmpss.h	CMPSS_getDACValueLow
<b>RAMPDLYA</b>	
cmpss.h	CMPSS_getRampDelayValue
<b>RAMPDLYS</b>	
cmpss.c	CMPSS_configRamp

**Table 16-25. CMPSS Registers to Driverlib Functions (continued)**

File	Driverlib Function
cmpss.h	CMPSS_setRampDelayValue
<b>CTRIPLFILCTL</b>	
cmpss.c	CMPSS_configFilterLow
cmpss.h	CMPSS_initFilterLow
<b>CTRIPLFILCLKCTL</b>	
cmpss.c	CMPSS_configFilterLow
<b>CTRIPHFILCTL</b>	
cmpss.c	CMPSS_configFilterHigh
cmpss.h	CMPSS_initFilterHigh
<b>CTRIPHFILCLKCTL</b>	
cmpss.c	CMPSS_configFilterHigh

## Sigma Delta Filter Module (SDFM)

---



---

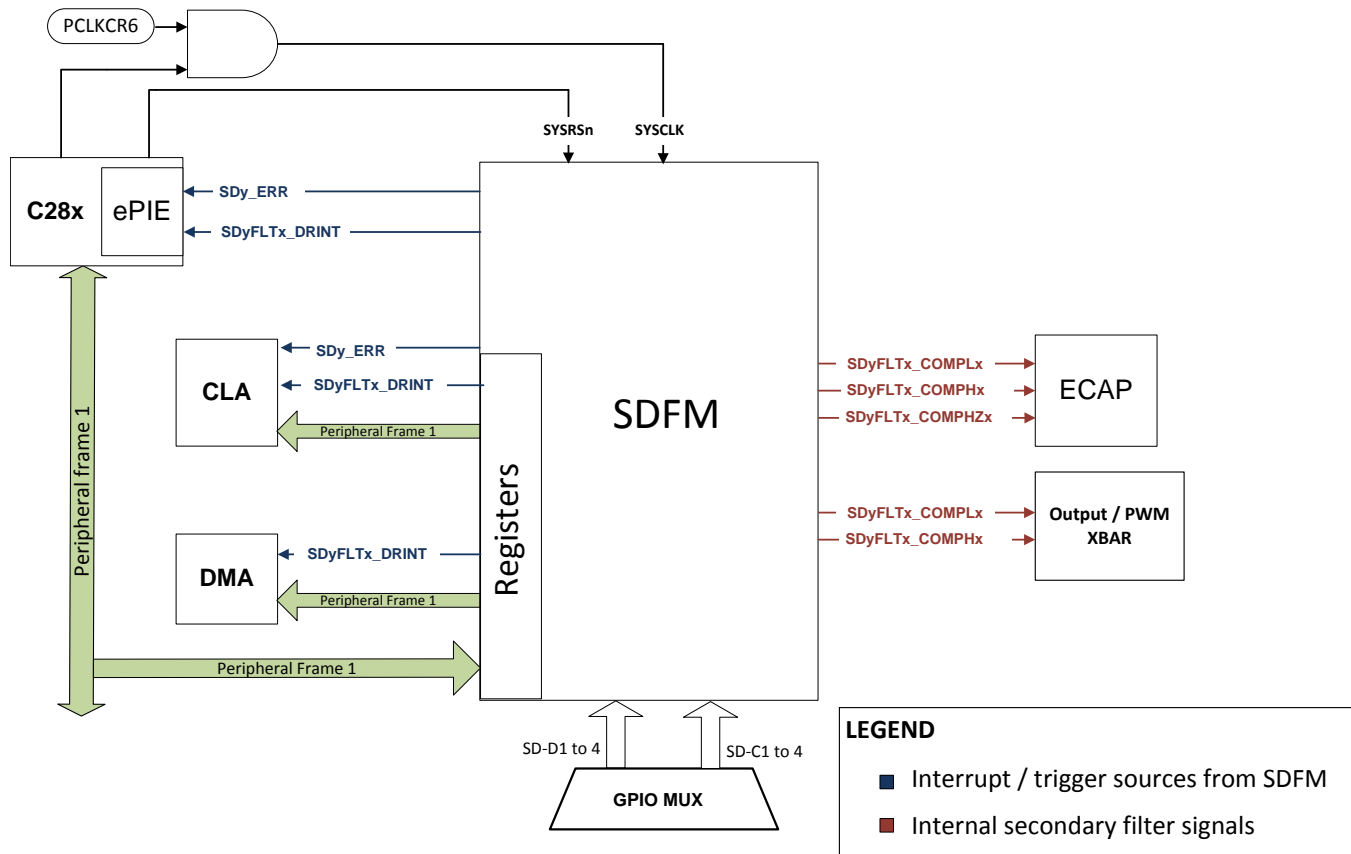
This chapter describes the sigma delta filter module (SDFM). The SDFM is a four-channel digital filter designed specifically for current measurement and resolver position decoding in motor control applications. Each input channel can receive an independent delta-sigma ( $\Delta\Sigma$ ) modulator bit stream. The bit streams are processed by four individually-programmable digital decimation filters. The filter set includes a fast comparator (secondary filter) for immediate digital threshold comparisons for over-current and under-current monitoring, and zeros crossing detection.

Topic	Page
17.1 Introduction .....	1726
17.2 Configuring Device Pins .....	1729
17.3 Input Control Unit .....	1730
17.4 Sinc Filter .....	1732
17.5 Data (Primary) Filter Unit .....	1734
17.6 Comparator (Secondary) Filter Unit.....	1738
17.7 Interrupt Unit .....	1740
17.8 SDFM Registers .....	1743

## 17.1 Introduction

Figure 17-1 shows the SDFM CPU Interface.

Figure 17-1. Sigma Delta Filter Module (SDFM) CPU Interface



### 17.1.1 SDFM Features

SDFM features include:

- Eight external pins per SDFM module
  - Four sigma-delta data input pins per SDFM module (SD-Dx, where x = 1 to 4)
  - Four sigma-delta clock input pins per SDFM module (SD-Cx, where x = 1 to 4)
- Four different configurable modulator clock modes:
  - Mode 0: Modulator clock rate equals the modulator data rate.
  - Mode 1: Modulator clock rate running at half the modulator data rate.
  - Mode 2: Modulator data is Manchester-encoded. Modulator clock is not required.
  - Mode 3: Modulator clock rate is double that of the modulator data rate
- Four independent, configurable secondary filter (comparator) units per SDFM module:
  - Four different filter type selection (Sinc1/Sinc2/Sincfast/Sinc3) options available
  - Ability to detect over-value condition, under-value condition, and Threshold-crossing conditions
  - OSR value for comparator filter unit (COSR) programmable from 1 to 32
- Four independent configurable primary filter (data filter) units per SDFM module:
  - Four different filter type selection (Sinc1/Sinc2/Sincfast/Sinc3) options available
  - OSR value for data filter unit (DOSR) programmable from 1 to 256
  - Ability to enable or disable (or both) individual filter module

- Ability to synchronize all four independent filters of an SDFM module by using the Master Filter Enable (MFE) bit or by using PWM signals
- Data filter output can be represented in either 16 bits or 32 bits.
- Data filter unit has a programmable mode FIFO to reduce interrupt overhead. The FIFO has the following features:
  - The primary filter (data filter) has a 16-deep x 32-bit FIFO.
  - The FIFO can interrupt the CPU after programmable number of data-ready events.
  - FIFO Wait-for-Sync feature: Ability to ignore data-ready events until the PWM synchronization signal (SDSYNC) is received. Once the SDSYNC event is received, the FIFO is populated on every data-ready event.
  - Data filter output can be represented in either 16 bits or 32 bits.
- PWMx.SOCA/SOCB can be configured to serve as SDSYNC source on a per-data-filter-channel basis.
- PWMs can be used to generate a modulator clock for sigma-delta modulators.

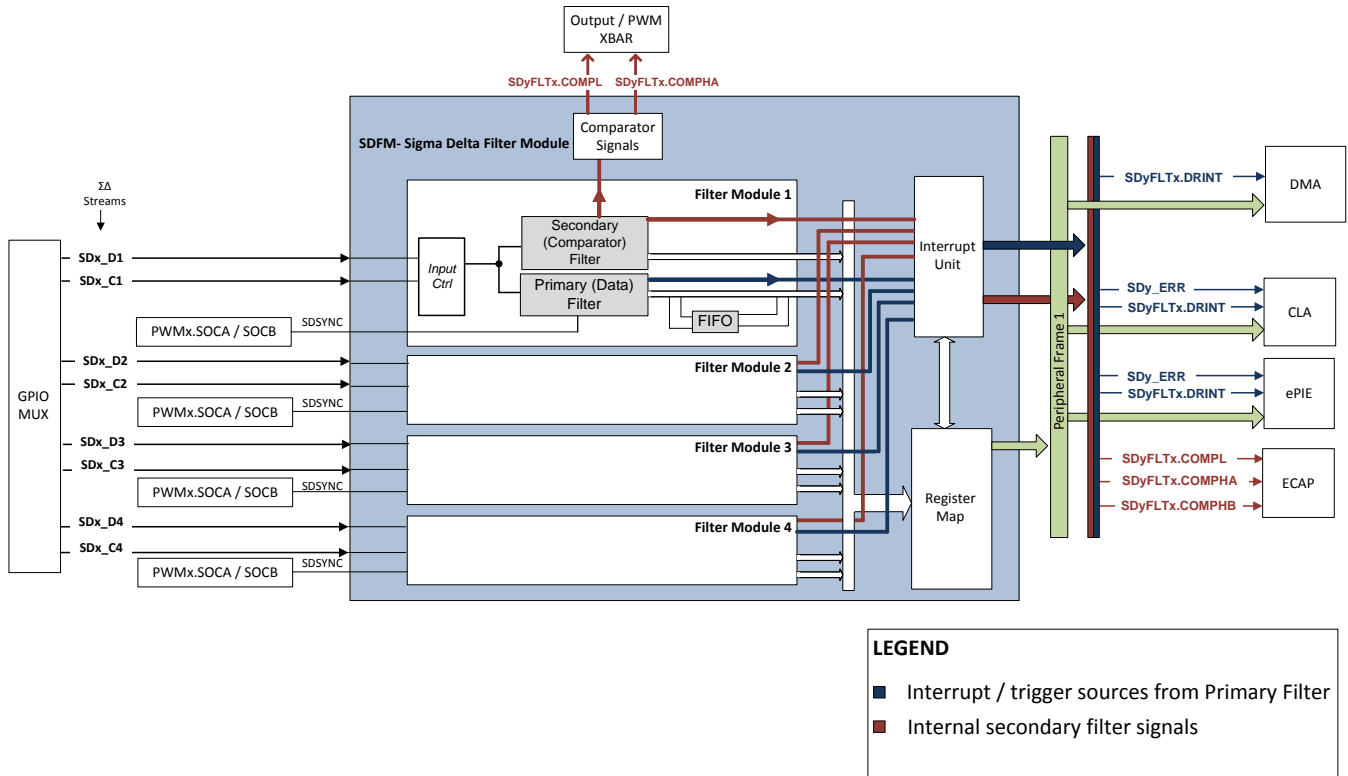
### 17.1.2 Block Diagram

Each SDFM module has four independent filter modules. These filter modules are identical and can be configured independently. Each individual filter module has the following units:

- Input control unit
- Primary filter (data filter) unit
- Secondary filter (comparator filter) unit with 4 independent comparators

[Figure 17-2](#) shows the SDFM module block diagram. The SDFM port operation is configured and controlled by the registers listed in .

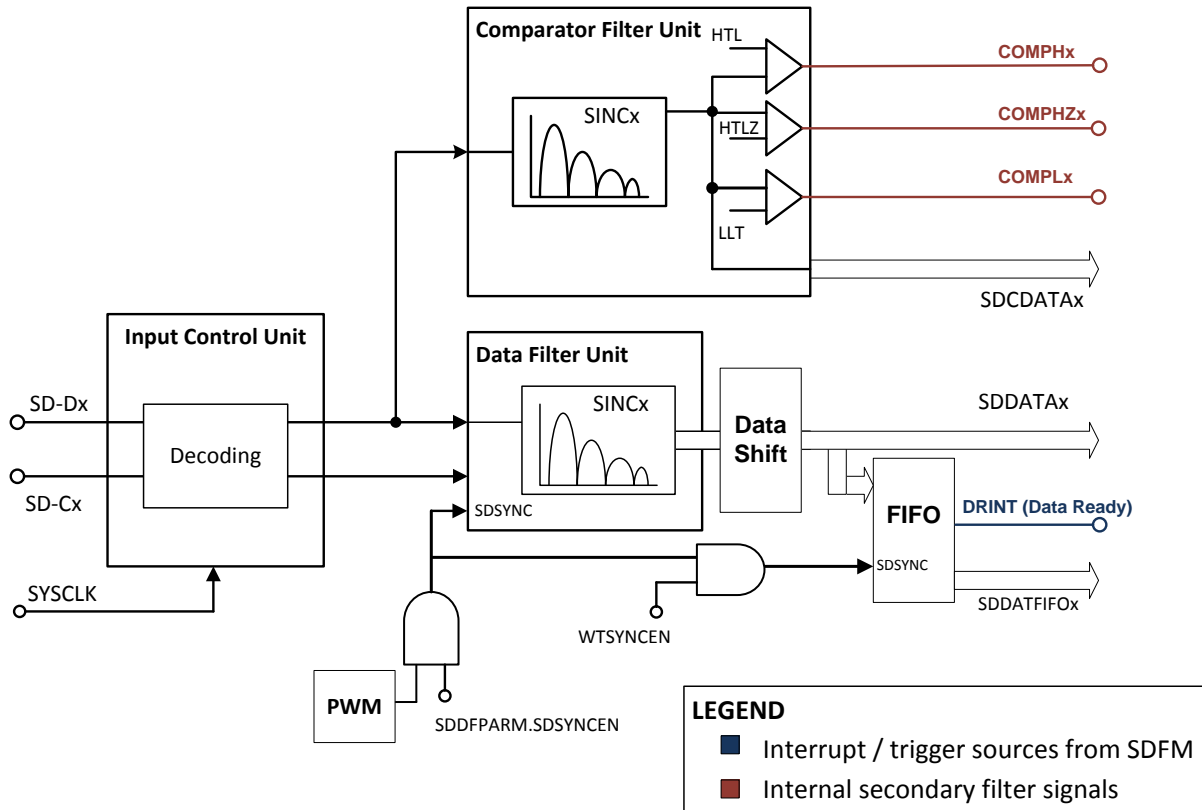
**Figure 17-2. Sigma Delta Filter Module (SDFM) Block Diagram**



Each filter module shown in [Figure 17-3](#) has a primary (data) filter and a secondary (comparator) filter pair which receives the same bit stream. Except for the input bit stream, both the primary and secondary filter are completely independent of each other. Each of these filter modules can be independently configured. So, in a SDFM module, there is a total of four primary filters and four secondary filters.



Figure 17-3. Block Diagram of One Filter Module



## 17.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

For proper SDFM operation, following GPIO input qualification can be used. Other GPIO qualifications are not supported.

- If GPIO Input qualification is ASYNC, please make sure to check SDFM Electrical Data and Timing (Using ASYNC) requirement is met and be aware of the warning message posted below.
- If GPIO Input qualification is SYNC, please make sure to check SDFM Electrical Data and Timing (Using SYNC) requirement is met and be aware of the warning message and note posted below.

### WARNING

The SDFM clock inputs (SDx\_Cy pins) directly clock the SDFM module. Any glitches or ringing noise on these inputs can corrupt the SDFM module operation. Special precautions should be taken on these signals to ensure a clean and noise-free signal that meets SDFM timing requirements. Precautions such as series termination for ringing due to any impedance mismatch of the clock driver and spacing of traces from other noisy signals are recommended.

**NOTE:** The SDFM Synchronized GPIO (SYNC) option provides protection against SDFM module corruption due to occasional random noise glitches on the SDx\_Cy pin that may result in a false comparator trip and filter output.

The SDFM Synchronized GPIO (SYNC) option does not provide protection against persistent violations of the above timing requirements. Timing violations will result in data corruption proportional to the number of bits which violate the requirements

See the *GPIO* chapter for more details on GPIO mux and settings.

### 17.3 Input Control Unit

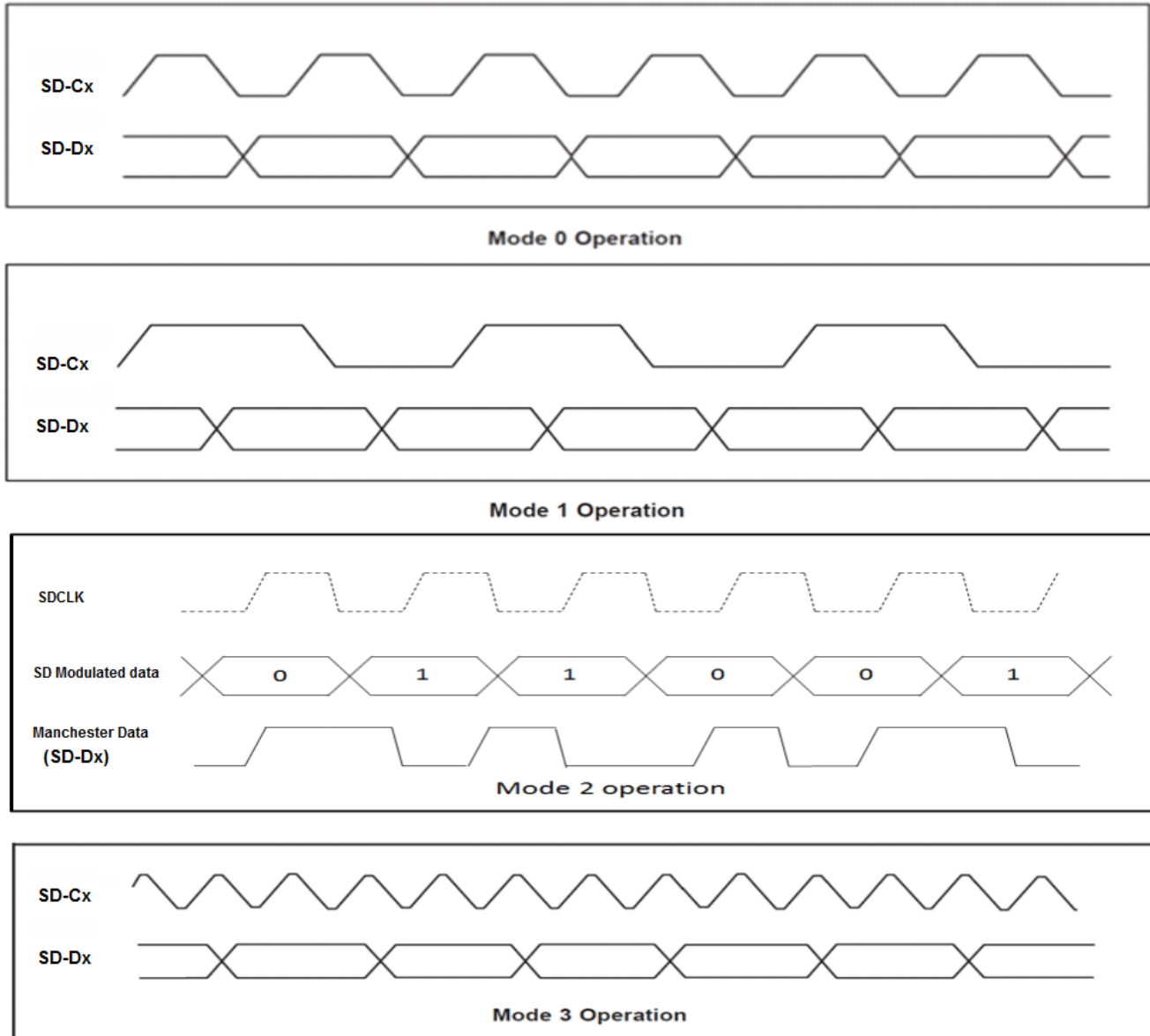
The input control unit receives sigma delta modulated data and a sigma delta modulated clock. The modulated data received is captured and passed on to the data filter unit and comparator unit. This unit can be configured to receive the modulated data in four different modes. [Table 17-1](#) and [Figure 17-4](#) show how SDCTLPARMx.MOD bits can be configured in these four different modes.

**Table 17-1. Modulator Clock Modes**

MODULATOR MODE [MOD]	DESCRIPTION
0	The modulator clock is running with the modulator data rate. The modulator data is strobed at every rising edge of the modulator clock.
1	The modulator clock is running with half of the modulator data rate. The modulator data is strobed at every edge of the modulator clock.
2	The modulator clock is off and the modulator data is Manchester-encoded.
3	The modulator clock is running with double the modulator data rate. The modulator data is strobed at every other positive modulator clock edge.

**NOTE:** In order to achieve the maximum value, the sigma-delta modulator has to be operated at absolute maximum positive or negative full scale, which is outside of the recommended full scale range of 80% of most sigma-delta modulators.

**Figure 17-4. Different Modulator Modes Supported**



When MOD=2, data and modulated clock signals are encoded into modulated data as shown in Mode 2 of [Figure 17-4](#). In this mode, the clock input SD-Cx pin should be left floating. The input control unit performs continuous automatic calibration to achieve optimum decoding performance.

## 17.4 Sinc Filter

Both comparator filter and data filter available in SDFM have the Sinc<sup>N</sup> filter as its core. The Sinc<sup>N</sup> filter is essentially a low pass filter which converts the input bit stream into digital data by digital filtering and decimation. This filtered digital data represents analog input given to the sigma delta modulator. Simplified Sinc<sup>N</sup> architecture consists of cascaded integrators and differentiators separated by a down-sampler as shown in [Figure 17-6](#). The Z-transfer function of the Sinc filter of order N is shown below.

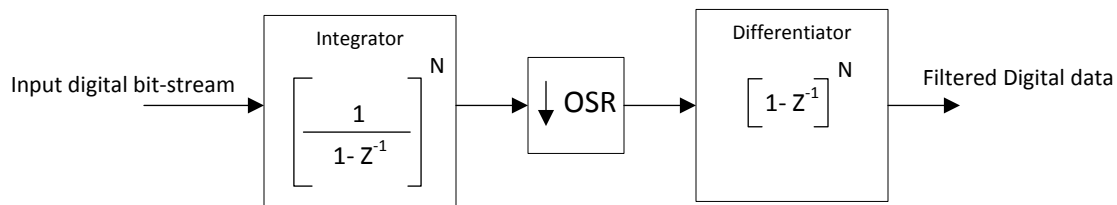
**Figure 17-5. Z-Transform of Sinc Filter of Order N**

$$H(Z) = \left[ \frac{1 - Z^{-OSR}}{1 - Z^{-1}} \right]^N$$

N = Order of Sinc filter

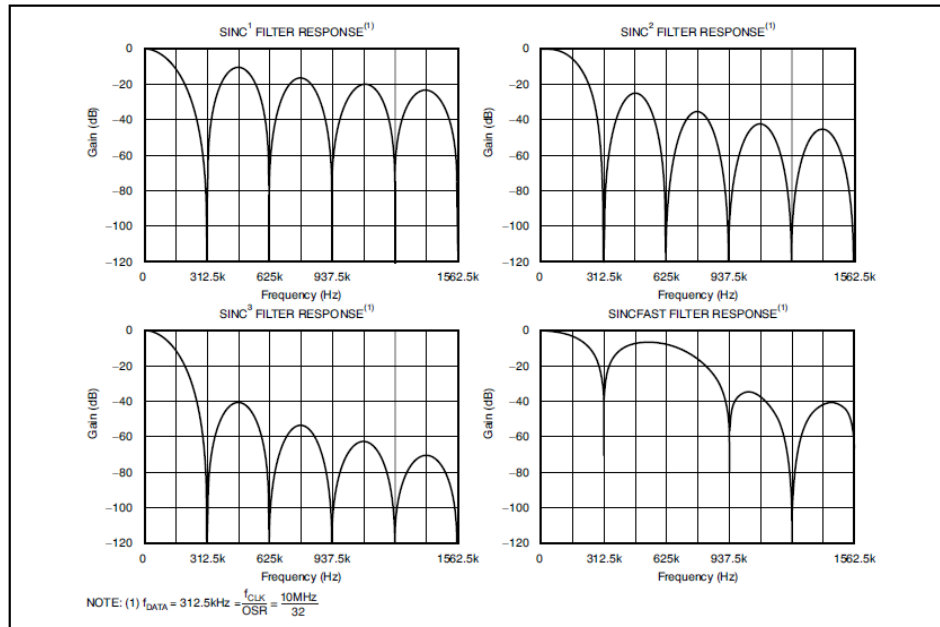
OSR = Over Sampling Ratio

**Figure 17-6. Simplified Sinc Filter Architecture**



Effective resolution of the Sinc filter (ENOB) depends upon filter type, OSR and sigma-delta modulator frequency. Typically, higher resolution (or) ENOB can be achieved by higher OSR for a given filter type; however, the tradeoff is increased filter delay. It is important to choose the right sigma delta modulator by studying the optimal speed versus resolution tradeoff. Refer to the corresponding sigma delta modulator datasheet to determine the effective resolution for a given Sinc filter configuration. The figure below shows the frequency response of different filter structures when OSR = 32 and when the sigma delta modulator frequency is 10 MHz.

Figure 17-7. Frequency Response of different Sinc Filters



The order of different sinc filter is shown in the table below.

Table 17-2. Order of Sinc Filter

filter Type	Order of Sinc Filter
Sinc1	1
Sinc2	2
Sinc3	3
SincFast	3

### 17.4.1 Data Rate and Latency of the Sinc Filter

The data rate of the sinc filter (filter throughput) represented in samples/sec can be calculated by the formula shown here:

$$\text{Data rate of Sinc filter} = \frac{\text{Modulator data rate}}{\text{OSR}}$$

The latency of the sinc filter represented in secs is defined as the amount of time taken by a sinc filter type to deliver the correct filtered output upon initiation. For a given filter type, latency can be calculated as shown in this equation:

$$\text{Latency of Sinc filter} = \frac{\text{Order of Sinc filter}}{\text{Data rate of Sinc filter}}$$

**Example configuration:**

- Sinc filter type = sinc3
- Modulator data rate = 10 MHz
- OSR = 256
- Data rate of Sinc Filter = 10 MHz / 256 = 39.1 K samples / sec
- Sinc filter latency = 76.8 μs

Sinc filter type	= sinc2
Modulator data rate	= 10 MHz
OSR	= 256
Data rate of Sinc Filter	= 10 MHz / 256 = 39.1 K samples / sec
Sinc filter latency	=51.2 $\mu$ s

## 17.5 Data (Primary) Filter Unit

The data filter is a configurable Sinc filter which supports the following filter types: Sinc1, Sinc2, Sinc3 and SincFast. The data filter OSR (DOSR) settings can be configured from 1 to 256 and is independent of the comparator filter. Effective resolution of the data filter (ENOB) depends upon Data filter type, DOSR, and sigma-delta modulator frequency. By default, the data filter is disabled and setting of SDDPARAMx.FEN = 1 enables the data filter. The data filter output is represented in 26-bit signed integer in two's complement format. This filter unit translates a low input signal as '-1' and a high input signal as '1'. The resulting calculation gives both positive and negative values for the output of the data filter. [Table 17-3](#) shows the different full scale values that the data filter can store using different OSRs.

See [Section 17.4.1](#) to understand how to calculate data rate and latency of data filter.

**Table 17-3. Peak Data Values for Different DOSR/Filter Combinations**

DOSR	Sinc1	Sinc2	Sinc3	Sincfast
x	x	x <sup>2</sup>	x <sup>3</sup>	2x <sup>2</sup>
4	-4 to 4	-16 to 16	-64 to 64	-32 to 32
8	-8 to 8	-64 to 64	-512 to 512	-128 to 128
16	-16 to 16	-256 to 256	-4096 to 4096	-512 to 512
32	-32 to 32	-1024 to 1024	-32,768 to 32,768	-2048 to 2048
64	-64 to 64	-4096 to 4096	-262,144 to 262,144	-8192 to 8192
128	-128 to 128	-16,384 to 16,384	-2,097,152 to 2,097,152	-32,768 to 32,768
256	-256 to 256	-65,536 to 65,536	-16,777,216 to 16,777,216	-131,072 to 131,072

### 17.5.1 32-bit or 16-bit Data Filter Output Representation

The data filter output can be represented in either 32-bit or 16-bit format.

#### 32-bit data filter representation:

- When SDDPARAMx.DR = 1, data filter output is represented in 32-bit format. Writes to shift control bits do not have any bearing on the output of the data filter in this configuration.

#### 16-bit data filter representation:

- By default, data filter output is represented in 16-bit format
- When SDDPARAMx.DR = 0, data filter output is represented in 16-bit format. But it is the responsibility of the user to configure the corresponding shift control bits in the SDDPARAMx register to control which 16-bit part of the 32-bit word is sent to the register map.

For example, for the data filter configuration below:

- Filter type = Sinc3
- OSR = 128
- SDDPARAMx.DR = 0

The data filter with a 26-bit signed output value can be in the range of -16,777,216 to 16,777,216. But, 16-bit signed output supports values only from -32,768 to 32,767. Therefore, it is required to configure shift control bits (SDDPARAMx.SH) to 7 to represent the data filter output correctly in 16-bit format. The table below shows the configuration settings of shift control bits for different OSR and filter types.

**Table 17-4. Shift Control Bit Configuration Settings**

OSR	SINC1	SINC2	SINCFAST	SINC3
1 to 31	0	0	0	0
32 to 40	0	0	0	1
41 to 50	0	0	0	2
51 to 63	0	0	0	3
64 to 80	0	0	0	4
81 to 101	0	0	0	5
102 to 127	0	0	0	6
128 to 161	0	0	1	7
162 to 181	0	0	1	8
182 to 203	0	1	2	8
204 to 255	0	1	2	9
256	0	2	3	10

### **WARNING**

**Configuring shift control bits incorrectly will result in getting incorrect 16-bit data filter output.**

## 17.5.2 Data FIFO

Each primary (data) filter channel has a 16-level deep, 32-bit FIFO.

FIFOs can be configured to collect a programmable number of data filter samples before issuing data-ready interrupt. This reduces the number of data-ready interrupts generated and resulting interrupt overhead for managed data flow.

By default, FIFO operation is disabled. FIFOs can be enabled by setting `SDFIFOCTLx.FFEN = 1`. When FIFO is enabled, each data-ready event from the data filter will populate the FIFO, and the status of the FIFO at any given time is updated in the `SDFIFOCTLx.SDFFST` bit field.

### Setting up FIFO to interrupt after receiving programmable number of data ready events:

- Enable SDFM FIFO (Set `SDFIFOCTLx.FFEN = 1`)
- Enable SDFM FIFO interrupt (Set `SDFIFOCTLx.FFIEN = 1`)
- Configure `SDFIFOCTLx.SDFFIL` bit field to any value between 0 to 16
- Configure SDFM data ready event to interrupt on FIFO interrupt (`SDFFINT`) (Set `SDFFINTx = 1`)
- Select data-ready interrupt source is `SDFFINTx` (`DRINTx = SDFFINTx`) (`SDFIFOCTLx.DRINTSEL = 1`)

When the `SDFIFOCTLx.SDFFST >= SDFIFOCTLx.SDFFIL` condition is met, the `SDIFLG.SDFFINTx` bit is set and an interrupt is generated on the `DRINTx`. `SDIFLG.SDFFINTx` flag can be cleared by setting the `SDIFLGCLR.SDFFINTx` bit field.

### Wait for Sync feature:

The FIFO wait for sync feature can be used to ignore data-ready events from the data filter until the `SDSYN` (from PWM) event is triggered.

By default, the Wait for Sync feature is disabled. This feature can be enabled by setting `SDSYNcx.WTSYNCEN = 1`

### When the wait for sync feature is disabled:

FIFOs get populated on every data ready event until the FIFO gets full (or) when `SDFIFOCTLx.SDFFST >= SDFIFOCTLx.SDFFIL`.

### When the wait for sync feature enabled:

FIFOs will not be get populated on every data ready event until it receives a SDSYNC event. On a SYSYNC event, it sets SDSYNCLx.WTSYNFLG = 1 and data ready events from primary filter will start populating FIFO until either the FIFOs get full (or) when SDFIFOCTLx.SDFFST >= SDFIFOCTLx.SDFFIL. WTSYNFLG can be cleared either automatically (or) manually.

When WTSYNFLG = 0, FIFOs contents are frozen and subsequent data ready events don't get populate FIFO until next SDSYNC event.

#### WTSYNFLG automatic clear mode:

By default, this mode is enabled. When SDSYNCLx.WTSCLEN = 1, WTSYNFLG is automatically cleared on SDFFINT event.

#### WTSYNFLG manual clear mode:

Setting SDSYNCLx.WTSYNCLR = 1 can be used to clear WTSYNFLG manually.

#### Clearing FIFO contents:

FIFO contents can cleared by any of the following methods:-

- Disabling FIFO clear FIFO contents. This can be done by clearing SDFIFOCTLx.FFEN = 0.
- Disabling Primary filter clear FIFO contents. This can done by either clearing SDDFPARMx.FEN = 0 (or) by clearing SDMFILEN.MFE = 0.
- FIFO contents can also be automatically cleared upon receiving the SDSYNC event. By default, this feature is disabled and this feature can be enabled by setting FIFO Clear-on-SDSYNC enable (SDSYNCLx.FFSYNCCLEN = 1).

**Note:** The above feature is only enabled when wait for sync feature is enabled (SDSYNCLx.WTSYNCCEN = 1).

#### FIFO debug access behavior:

Debug access of SDDATFIFOx registers does not affect FIFO pointers. On a CPU / CLA / DMA access to SDDATFIFOx register, FIFO read pointers would be advanced to next available entry in FIFO.

### 17.5.3 SDSYNC Event

Primary (data) filters can be synchronized with respect to the PWM event (called SDSYNC event). The SDSYNC signal from the PWM module is used to reset the DOSR counter. This feature is by default disabled and can be enabled by setting SDDFPARMx.SDSYNCCEN = 1. Each primary filter can be synchronized from any of the available PWMx. SOCA / SOCB signals. (see [Table 17-5](#)). The SDSYNCLx.SDSYNCCSEL bits allow the user to configure which PWM signal provides the SDSYNC pulse to the primary filter.

**Table 17-5. SDSYNCLx.SDSYNCCSEL**

SDSYNCLx.SDSYNCCSEL[1:0]				
SDSYNCCSEL[5:2]	0	1	2	3
0	PWM1.SOCA	PWM1.SOCB	Rsvd	Rsvd
1	PWM2.SOCA	PWM2.SOCB	Rsvd	Rsvd
2	PWM3.SOCA	PWM3.SOCB	Rsvd	Rsvd
3	PWM4.SOCA	PWM4.SOCB	Rsvd	Rsvd
4	PWM5.SOCA	PWM5.SOCB	Rsvd	Rsvd
5	PWM6.SOCA	PWM6.SOCB	Rsvd	Rsvd
6	PWM7.SOCA	PWM7.SOCB	Rsvd	Rsvd
7	PWM8.SOCA	PWM8.SOCB	Rsvd	Rsvd
8	Rsvd	Rsvd	Rsvd	Rsvd
9	Rsvd	Rsvd	Rsvd	Rsvd
10	Rsvd	Rsvd	Rsvd	Rsvd
11	Rsvd	Rsvd	Rsvd	Rsvd
12	Rsvd	Rsvd	Rsvd	Rsvd

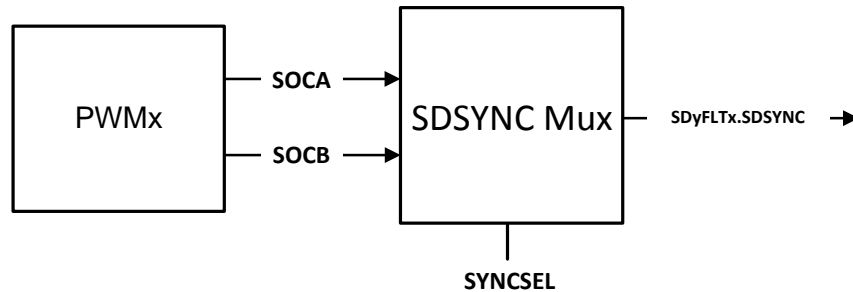


**Table 17-5. SDSYNcx.SYNCSEL (continued)**

SDSNYx.SYNCSEL[1:0]				
13	Rsvd	Rsvd	Rsvd	Rsvd
14	Rsvd	Rsvd	Rsvd	Rsvd
15	Rsvd	Rsvd	Rsvd	Rsvd

Figure 17-8 shows how the PWM signals are connected to SDFM.

**Figure 17-8. SDSYNC Event**



**NOTE:** Ensure that ONLY ONE SDSYNC event will be generated per PWM timer period. Using PWM in up-count or down-count mode would automatically ensure that you get ONLY SDSYNC event. But, if up-down count mode is used, then make sure that only one SDSYNC event per PWM cycle is generated; otherwise, the filter synchronizer will corrupt SDFM timing by providing two pulses per PWM cycle.

Because of the inherent architecture of the Sinc filter (Sinc1, Sinc2, Sinc3, SincFast), the first few samples, depending upon filter type will be incorrect. Table 17-6 shows the number of incorrect samples on the following conditions:-

- When Sinc filter is enabled / configured for first time.
- When Sinc filter is disabled / re-enabled or reconfigured in the middle of operation.
- When data filter receives SDSYNC event from PWM.

**Table 17-6. Number of Incorrect Samples Tabulated**

Filter type	Number of incorrect samples after the filter is enabled and configured
Sinc1	No incorrect sample
Sinc2	The first sample of the Sinc2 filter is incorrect
SincFast	The first two samples of the SincFast filter are incorrect
Sinc3	The first two samples of the Sinc3 filter are incorrect

## WARNING

**SDFM comparator interrupts should be enabled only after providing sufficient settling time to make sure the comparator filter does not trip on these incorrect samples. Therefore, SDFM comparator interrupts (IELx and IEHx) should be enabled only after a sufficient delay is provided after the comparator filter is configured. This sufficient delay is calculated by adding the latency of the comparator filter and five SD-Cx clock cycles.**

### 17.6 Comparator (Secondary) Filter Unit

Most control systems require protection of the system by tripping the PWM in case the current or voltage goes out of bounds. The primary purpose of the secondary (comparator) filter is to allow the user to monitor input conditions with a fast settling time. This allows the user to trip PWMs to protect the system from potential damage.

---

**NOTE:** The secondary (comparator) filter cannot be synchronized with respect to the PWM event (SDSYNC event).

---

The comparator filter is a configurable Sinc filter which supports the following filter types: - Sinc1, Sinc2, Sinc3 and SincFast. The comparator OSR (COSR) settings can be configured from 1 to 32 and is independent of the data filter. Effective resolution of the comparator filter (ENOB) depends upon the comparator filter type, COSR, and sigma-delta modulator frequency. By default, the comparator filter is disabled and setting SDCPARMx.CEN = 1 enables the comparator filter. The comparator filter output is represented in 16-bit unsigned format. This filter unit translates a low input signal as '0' and a high input signal as '1'. The resulting calculations give only positive values for the output of the comparator filter. [Table 17-7](#) shows the different full-scale values that the comparator filter can store using different OSRs.

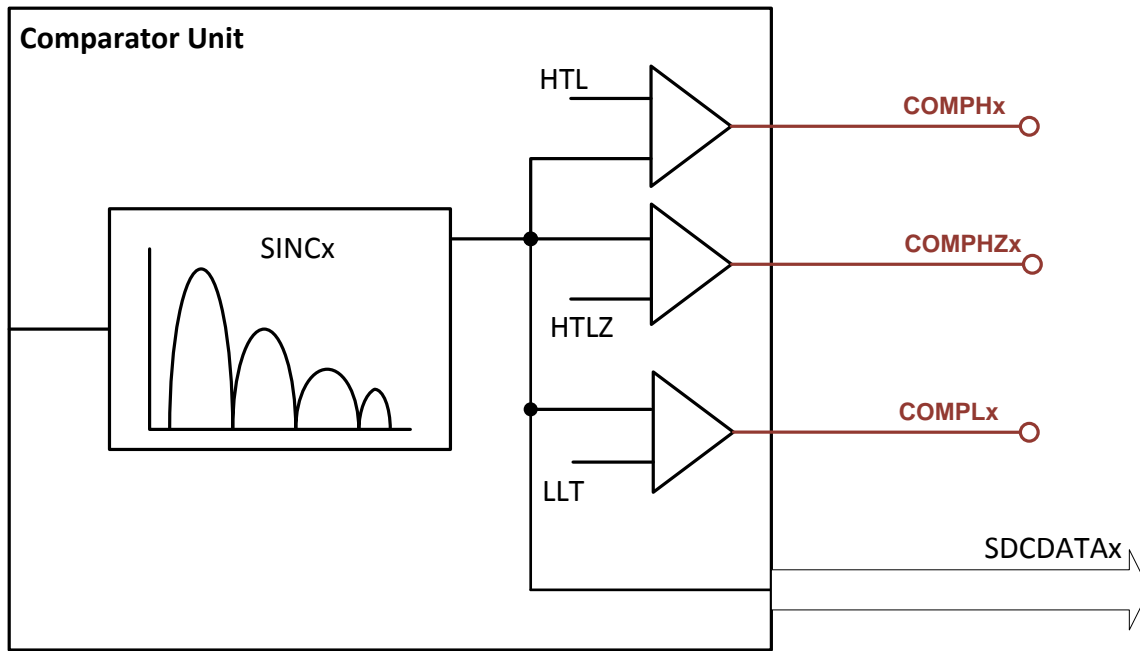
**Table 17-7. Peak Data Values for Different OSR/Filter Combinations**

OSR	Sinc1	Sinc2	Sinc3	Sincfast
x	0 to x	0 to x2	0 to x3	0 to 2x2
4	0 to 4	0 to 16	0 to 64	0 to 32
8	0 to 8	0 to 64	0 to 512	0 to 128
16	0 to 16	0 to 256	0 to 4096	0 to 512
32	0 to 32	0 to 1024	0 to 32,768	0 to 2048

See [Section 17.4.1](#) to understand how to calculate data rate and latency of comparator filter.

The output of the comparator filter is memory-mapped and can be read in the SDCDATAx register. This register, SDCDATAx, is updated every COSR number of SD-Cx cycles. The comparator filter digital output is connected to digital comparators explained below.

**Figure 17-9. Comparator Unit Structure**



### 17.6.1 Higher threshold (HLT) comparators

- High threshold comparator can be used to detect over-value condition.
- When comparator data  $\geq$  higher threshold register, a high threshold event is generated.
- Higher threshold comparator events except for COMPHzx can be configured to trigger following events : CPU interrupt, CLA task, PWM trip.
- Higher threshold comparator events can be used in conjunction with ECAP to measure the frequency / duty cycle of threshold crossing
- This device has two High Threshold comparators.
  - **Higher Threshold (HLT) comparator:**
    - The high threshold register (SDCMPHx.HLT) can be used to configure this comparator.
    - When SDCTL.MIE = 1 and SDCPARMx.IEH = 1, a high threshold event triggers an SDFM interrupt (SDINT) and sets the SDIFLG.IEHx flag showing an over-value condition is triggered. This SDIFLG.IEHx flag can be reset if the corresponding bit in the SDIFLGCLR register is set and if the interrupt source is no longer active.
  - **Higher Threshold (HTLZ) comparator :**
    - The high threshold Register (SDCMPHzx.HTLZ) can be used to configure this comparator.
    - When comparator data  $\geq$  SDCMPHzx.HTLZ, it can generate a higher threshold (B) event (COMPHzx) and sets the corresponding SDSTATUS.HZx flag. But, this event cannot be configured to generate an SDFM interrupt (SDINT). The high threshold (HTLZ) comparator can be used in conjunction with ECAP to measure the frequency / duty cycle of Threshold crossing events.

#### Example

To measure the frequency / duty cycle of zero crossing event, the user needs to configure SDCMPHzx.HTLZ = 0x4000 and enable High level (B) Threshold crossing (SDCPARMx.HZEN =

1). The ECCTL0.INPUTSEL register bit field should also be configured to pass on the COMPHZx signal to eCAP for frequency / duty cycle measurement.

### 17.6.2 Lower Threshold (LLT) comparators

- The low threshold comparator can be used to detect under-value condition.
- When comparator data  $\leq$  Lower Threshold register, a low threshold event is generated.
- Lower threshold comparator events can be configured to trigger following events : CPU interrupt, CLA task, PWM trip.
- Lower threshold comparator events can be used in conjunction with ECAP to measure the frequency / duty cycle of Threshold crossing
- This device has one low threshold comparators.
  - **Lower Threshold (LLT) Comparator** : When SDCTL.MIE = 1 and SDCPARAMx.IEL = 1, the low threshold event triggers an SDFM interrupt (SDINT) and sets the SDIFLG.IELx flag showing an under-value condition is triggered. This SDIFLG.IELx flag can be reset if the corresponding bit in the SDIFLGCLR register is set and if the interrupt source is no longer active.

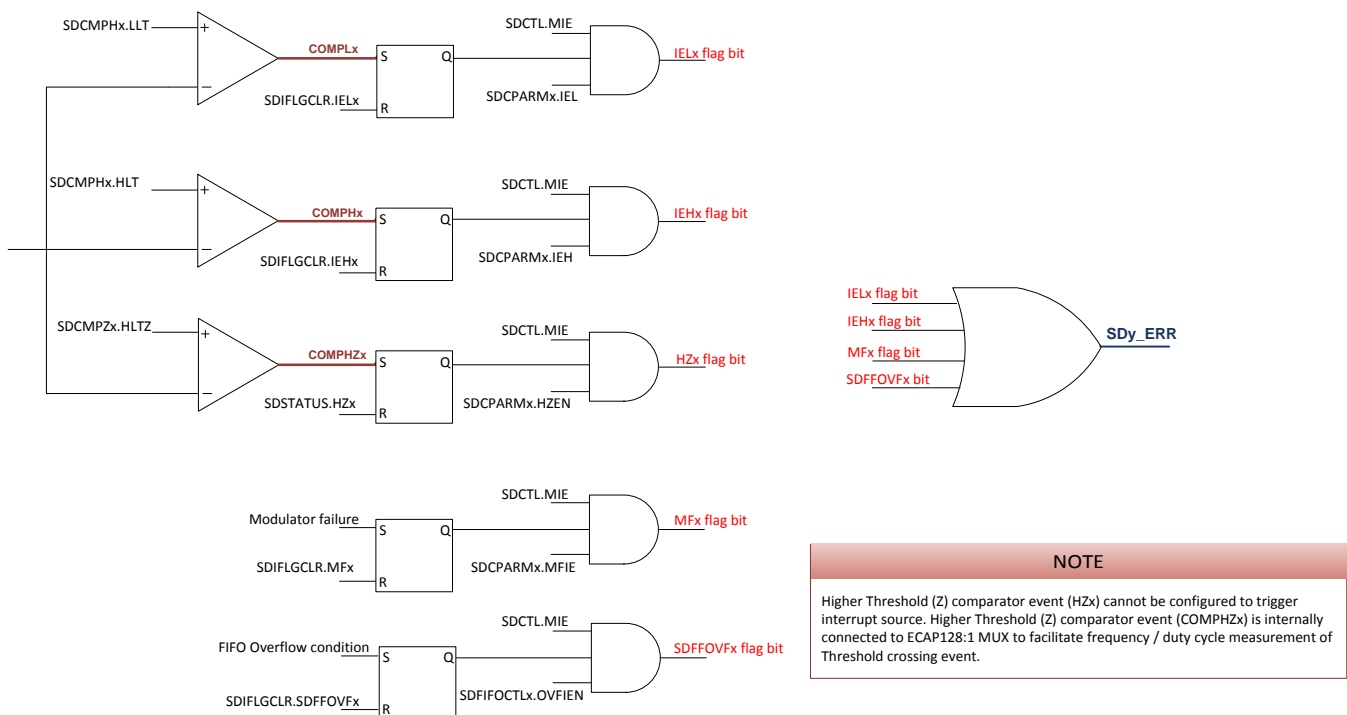
## 17.7 Interrupt Unit

Each SDFM can generate five CPU interrupts such as SDFM Error (SDy\_ERR) and SDFM data ready (SDy\_DRINT1 / SDy\_DRINT2, SDy\_DRINT3, SDy\_DRINT4) interrupts for each filter module.

### 17.7.1 SDFM (SDyERR) Interrupt sources:

Figure 17-10 shows the structure of SDy\_ERR interrupt. SDy\_ERR interrupt can be triggered by any of these 16 events.

Figure 17-10. SDFM Error (SD\_ERR) interrupt sources



#### 1. Comparator Lower Threshold events (COMPLx)

COMPL events from any of the four comparator filter module can trigger CPU interrupt. This event can be configured to trigger SDy\_ERR interrupt only if below configurations are made:

- Enable master interrupt enable (SDCTL.MIE = 1)

- Enable comparator filter lower threshold interrupt (SDCPARMx.IEL = 1)

On a COMPL event, SDIFLG.IELx flag bit is set. This flag bit can only be reset if the corresponding bit in SDIFLGCLR register is set and if the interrupt source is no longer active.

## 2. Comparator High Threshold events (COMPHx)

COMPH events from any of the four comparator filter module can trigger CPU interrupt. This event can be configured to trigger SDy\_ERR interrupt only if below configurations are made:

- Enable master interrupt enable (SDCTL.MIE = 1)
- Enable comparator filter lower threshold interrupt (SDCPARMx.IEH = 1)

On a COMPH event, SDIFLG.IEHx flag bit is set. This flag bit can only be reset if the corresponding bit in SDIFLGCLR register is set and if the interrupt source is no longer active.

## 3. Modulator Failure (MFx) event

Modulator failures (MFx) are generated when SD-Cx goes missing. The modulator clock is considered missing if SD-Cx does not toggle for 64-SYSCLKs. MFx events from any of the four filter modules can trigger CPU interrupt. This event can be configured to trigger SDy\_ERR interrupt only if below configurations are made:

- Enable master Interrupt Enable (SDCTL.MIE = 1)
- Enable modulator clock failure interrupt source (SDCPARMx.MFIE = 1)

On a MFx event, SDIFLG.MFx flag bit is set. This flag bit can only be reset if the corresponding bit in SDIFLGCLR register is set and if the interrupt source is no longer active.

## 4. FIFO overflow (SDFFOVx) event

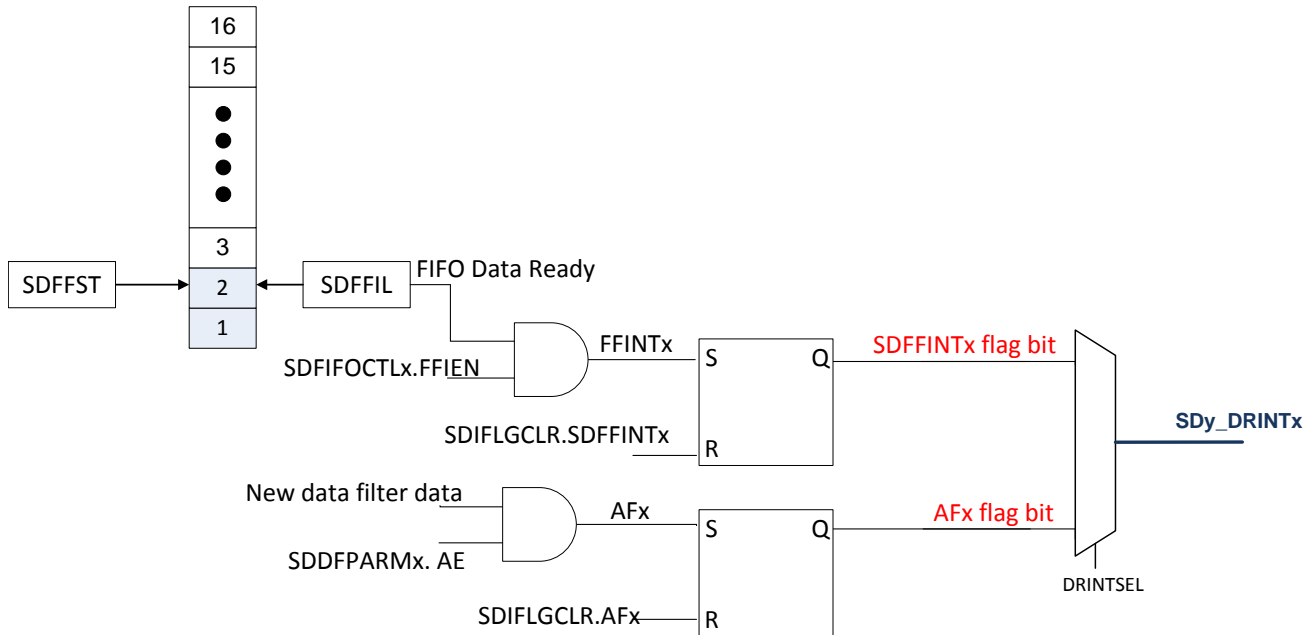
The number of filter data available in FIFO at any given point can be tracked in SDFIFOCTLx.SDFFST. If the number of words received in FIFO is greater than Max FIFO depth (16), SDFFOVx event is generated. SDFFOVx events from any of the four filter modules can trigger CPU interrupt. This event can be configured to trigger SDy\_ERR interrupt only if below configurations are made:

- Enable SDFM FIFO (Set SDFIFOCTLx.FFEN = 1)
- Enable SDFM FIFO overflow interrupt (Set SDFIFOCTLx.OVFIEN = 1) and
- Enable Master interrupt enable (Set SDCTL.MIE = 1)

On a SDFFOVx event, all subsequent data (primary) filter data is lost and is not stored in FIFO. SDIFLG.SDFFOVx flag bit is set on a FIFO overflow event and this bit can be cleared if the corresponding bit in SDIFLGCLR register is set and if the interrupt source is no longer active.

### 17.7.2 Data Ready (DRINT) Interrupt sources:

[Figure 17-11](#) shows the structure of interrupt SDy\_DRINTx interrupt. Each SDy\_DRINTx interrupt is triggered by corresponding Data Filter channel.

**Figure 17-11. SDFM Data Ready (SDy\_DRINTx) Interrupt**


### 1. Data Acknowledge (AFx)

When the primary filter is ready with a new filter data, AFx event is generated. AFx events from each filter can generate its own SDy\_DRINTx interrupt. This event can be configured to trigger SDy\_DRINTx interrupt only if below configurations are made:

- Enable individual filter interrupts (SDDFPARAMx. AE = 1)
- Select data-ready interrupt source AFx (DRINTx = AFx) (SDFIFOCTLx.DRINTSEL = 0)

On an AFx event, the SDIFLG.AFx flag bit is set. This flag bit can only be reset if the corresponding bit in SDIFLGCLR register is set and if the interrupt source is no longer active.

### 2. Four FIFO Data ready interrupt (SDFFINTx)

FIFO Data Ready event is generated whenever SDFIFOCTLx.SDFFST  $\geq$  SDFIFOCTLx.SDFFIL condition is met. FIFO data ready events from each filter can generate its own SDy\_DRINTx interrupt. This event can be configured to trigger SDy\_DRINTx interrupt only if below configurations are made:

Table 17-8 shows how the DRINTx output is selected.

- Enable SDFM FIFO (Set SDFIFOCTLx.FFEN = 1) and
- Enable SDFM FIFO interrupt (Set SDFIFOCTLx.FFIEN = 1)
- Select data-Ready interrupt source is SDFFINTx (DRINTx = SDFFINTx) (SDFIFOCTLx.DRINTSEL = 1)

**Table 17-8. SDFM Data-Ready Interrupt (SDy\_DRINTx) Output Selection**

DRINTSEL	AE	FFIEN	FFEN	DRINTx
0	0	x	X	0
0	1	x	X	AFx
1	x	0	X	0
1	x	x	0	0
1	x	1	1	SDFFINTx

## 17.8 SDFM Registers

This section describes the Sigma Delta Filter registers.

### 17.8.1 SDFM Base Addresses

**Table 17-9. SDFM Base Address Table**

Device Registers	Register Name	Start Address	End Address
Sdfm1Regs	SDFM_REGS	0x0000_5E00	0x0000_5E7F

## 17.8.2 SDFM\_REGS Registers

Table 17-10 lists the SDFM\_REGS registers. All register offset addresses not listed in Table 17-10 should be considered as reserved locations and the register contents should not be modified.

**Table 17-10. SDFM\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	SDIFLG	SD Interrupt Flag Register		<a href="#">Go</a>
2h	SDIFLGCLR	SD Interrupt Flag Clear Register		<a href="#">Go</a>
4h	SDCTL	SD Control Register	EALLOW	<a href="#">Go</a>
6h	SDMFILEN	SD Master Filter Enable	EALLOW	<a href="#">Go</a>
7h	SDSTATUS	SD Status Register		<a href="#">Go</a>
10h	SDCTLPARM1	Control Parameter Register for Ch1	EALLOW	<a href="#">Go</a>
11h	SDDFPARM1	Data Filter Parameter Register for Ch1	EALLOW	<a href="#">Go</a>
12h	SDDPARAM1	Data Parameter Register for Ch1	EALLOW	<a href="#">Go</a>
13h	SDCMPH1	High-level Threshold Register for Ch1	EALLOW	<a href="#">Go</a>
14h	SDCMPL1	Low-level Threshold Register for Ch1	EALLOW	<a href="#">Go</a>
15h	SDCPARM1	Comparator Filter Parameter Register for Ch1	EALLOW	<a href="#">Go</a>
16h	SDDATA1	Data Filter Data Register (16 or 32bit) for Ch1		<a href="#">Go</a>
18h	SDDATFIFO1	Filter Data FIFO Output(32b) for Ch1		<a href="#">Go</a>
1Ah	SDCDATA1	Comparator Filter Data Register (16b) for Ch1		<a href="#">Go</a>
1Ch	SDCMPHZ1	High-level (Z) Threshold Register for Ch1	EALLOW	<a href="#">Go</a>
1Dh	SDFIFOCTL1	FIFO Control Register for Ch1	EALLOW	<a href="#">Go</a>
1Eh	SDSYNC1	SD Filter Sync control for Ch1	EALLOW	<a href="#">Go</a>
20h	SDCTLPARM2	Control Parameter Register for Ch2	EALLOW	<a href="#">Go</a>
21h	SDDFPARM2	Data Filter Parameter Register for Ch2	EALLOW	<a href="#">Go</a>
22h	SDDPARAM2	Data Parameter Register for Ch2	EALLOW	<a href="#">Go</a>
23h	SDCMPH2	High-level Threshold Register for Ch2	EALLOW	<a href="#">Go</a>
24h	SDCMPL2	Low-level Threshold Register for Ch2	EALLOW	<a href="#">Go</a>
25h	SDCPARM2	Comparator Filter Parameter Register for Ch2	EALLOW	<a href="#">Go</a>
26h	SDDATA2	Data Filter Data Register (16 or 32bit) for Ch2		<a href="#">Go</a>
28h	SDDATFIFO2	Filter Data FIFO Output(32b) for Ch2		<a href="#">Go</a>
2Ah	SDCDATA2	Comparator Filter Data Register (16b) for Ch2		<a href="#">Go</a>
2Ch	SDCMPHZ2	High-level (Z) Threshold Register for Ch2	EALLOW	<a href="#">Go</a>
2Dh	SDFIFOCTL2	FIFO Control Register for Ch2	EALLOW	<a href="#">Go</a>
2Eh	SDSYNC2	SD Filter Sync control for Ch2	EALLOW	<a href="#">Go</a>
30h	SDCTLPARM3	Control Parameter Register for Ch3	EALLOW	<a href="#">Go</a>
31h	SDDFPARM3	Data Filter Parameter Register for Ch3	EALLOW	<a href="#">Go</a>
32h	SDDPARAM3	Data Parameter Register for Ch3	EALLOW	<a href="#">Go</a>
33h	SDCMPH3	High-level Threshold Register for Ch3	EALLOW	<a href="#">Go</a>
34h	SDCMPL3	Low-level Threshold Register for Ch3	EALLOW	<a href="#">Go</a>
35h	SDCPARM3	Comparator Filter Parameter Register for Ch3	EALLOW	<a href="#">Go</a>
36h	SDDATA3	Data Filter Data Register (16 or 32bit) for Ch3		<a href="#">Go</a>
38h	SDDATFIFO3	Filter Data FIFO Output(32b) for Ch3		<a href="#">Go</a>
3Ah	SDCDATA3	Comparator Filter Data Register (16b) for Ch3		<a href="#">Go</a>
3Ch	SDCMPHZ3	High-level (Z) Threshold Register for Ch3	EALLOW	<a href="#">Go</a>
3Dh	SDFIFOCTL3	FIFO Control Register for Ch3	EALLOW	<a href="#">Go</a>
3Eh	SDSYNC3	SD Filter Sync control for Ch3	EALLOW	<a href="#">Go</a>
40h	SDCTLPARM4	Control Parameter Register for Ch4	EALLOW	<a href="#">Go</a>
41h	SDDFPARM4	Data Filter Parameter Register for Ch4	EALLOW	<a href="#">Go</a>
42h	SDDPARAM4	Data Parameter Register for Ch4	EALLOW	<a href="#">Go</a>



**Table 17-10. SDFM\_REGS Registers (continued)**

Offset	Acronym	Register Name	Write Protection	Section
43h	SDCMPH4	High-level Threshold Register for Ch4	EALLOW	<a href="#">Go</a>
44h	SDCMPL4	Low-level Threshold Register for Ch4	EALLOW	<a href="#">Go</a>
45h	SDCPARM4	Comparator Filter Parameter Register for Ch4	EALLOW	<a href="#">Go</a>
46h	SDDATA4	Data Filter Data Register (16 or 32bit) for Ch4		<a href="#">Go</a>
48h	SDDATFIFO4	Filter Data FIFO Output(32b) for Ch4		<a href="#">Go</a>
4Ah	SDCDATA4	Comparator Filter Data Register (16b) for Ch4		<a href="#">Go</a>
4Ch	SDCMPHZ4	High-level (Z) Threshold Register for Ch4	EALLOW	<a href="#">Go</a>
4Dh	SDFIFOCTL4	FIFO Control Register for Ch4	EALLOW	<a href="#">Go</a>
4Eh	SDSYNC4	SD Filter Sync control for Ch4	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 17-11](#) shows the codes that are used for access types in this section.

**Table 17-11. SDFM\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R=0	R	Read
<b>Write Type</b>		
W	W	Write
W=1	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**17.8.2.1 SDIFLG Register (Offset = 0h) [reset = 0h]**

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

Return to the [Summary Table](#).

SD Interrupt Flag Register

**Figure 17-12. SDIFLG Register**

31	30	29	28	27	26	25	24
MIF	RESERVED						
R-0h							
23	22	21	20	19	18	17	16
SDFINT4	SDFINT3	SDFINT2	SDFINT1	SFFOV4	SFFOV3	SFFOV2	SFFOV1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
AF4	AF3	AF2	AF1	MF4	MF3	MF2	MF1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
IFL4	IFH4	IFL3	IFH3	IFL2	IFH2	IFL1	IFH1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 17-12. SDIFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MIF	R	0h	Set whenever any "error" interrupt (MF1-4,IFL1-4,IFH1-4,SDFFOVF1-4) is active Reset type: SYSRSn
30-24	RESERVED	R=0	0h	Reserved
23	SDFINT4	R	0h	SDFIFO data ready interrupt for Ch4 Reset type: SYSRSn
22	SDFINT3	R	0h	SDFIFO data ready interrupt for Ch3 Reset type: SYSRSn
21	SDFINT2	R	0h	SDFIFO data ready interrupt for Ch2 Reset type: SYSRSn
20	SDFINT1	R	0h	SDFIFO data ready interrupt for Ch1 0: SDFIFO data ready interrupt has NOT occurred 1: SDFIFO data ready interrupt has occurred Reset type: SYSRSn
19	SDFFOVF4	R	0h	FIFO Overflow Flag for Ch4 Reset type: SYSRSn
18	SDFFOVF3	R	0h	FIFO Overflow Flag for Ch3 Reset type: SYSRSn
17	SDFFOVF2	R	0h	FIFO Overflow Flag for Ch2 Reset type: SYSRSn
16	SDFFOVF1	R	0h	FIFO Overflow Flag for Ch1 0 - FIFO has not overflowed 1 - FIFO overflowed. # words received in FIFO > FIFO depth (16), NEW word is lost Reset type: SYSRSn

**Table 17-12. SDIFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	AF4	R	0h	Acknowledge flag for Filter 4 0: No new data available for Filter (in non-FIFO mode) 1: New data available for Filter (in non-FIFO mode) Reset type: SYSRSn
14	AF3	R	0h	Acknowledge flag for Filter 3 0: No new data available for Filter (in non-FIFO mode) 1: New data available for Filter (in non-FIFO mode) Reset type: SYSRSn
13	AF2	R	0h	Acknowledge flag for Filter 2 0: No new data available for Filter (in non-FIFO mode) 1: New data available for Filter (in non-FIFO mode) Reset type: SYSRSn
12	AF1	R	0h	Acknowledge flag for Filter 1 0: No new data available for Filter (in non-FIFO mode) 1: New data available for Filter (in non-FIFO mode) Reset type: SYSRSn
11	MF4	R	0h	Modulator Failure for Filter 4 0: Modulator is operating normally for Filter 1: Modulator failure for Filter Reset type: SYSRSn
10	MF3	R	0h	Modulator Failure for Filter 3 0: Modulator is operating normally for Filter 1: Modulator failure for Filter Reset type: SYSRSn
9	MF2	R	0h	Modulator Failure for Filter 2 0: Modulator is operating normally for Filter 1: Modulator failure for Filter Reset type: SYSRSn
8	MF1	R	0h	Modulator Failure for Filter 1 0: Modulator is operating normally for Filter 1: Modulator failure for Filter Reset type: SYSRSn
7	IFL4	R	0h	Low-level Interrupt flag for Ch4 0: Comparator filter output > SDCMPL4.LLT 1: Comparator filter output <= SDCMPL4.LLT Reset type: SYSRSn
6	IFH4	R	0h	High-level Interrupt flag for Ch4 0: Comparator filter output < SDCMPH4.HLT 1: Comparator filter output >= SDCMPH4.HLT Reset type: SYSRSn

**Table 17-12. SDIFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	IFL3	R	0h	Low-level Interrupt flag for Ch3 0: Comparator filter output > SDCMPL3.LLT 1: Comparator filter output <= SDCMPL3.LLT Reset type: SYSRSn
4	IFH3	R	0h	High-level Interrupt flag for Ch3 0: Comparator filter output < SDCMPH3.HLT 1: Comparator filter output >= SDCMPH3.HLT Reset type: SYSRSn
3	IFL2	R	0h	Low-level Interrupt flag for Ch2 0: Comparator filter output > SDCMPL2.LLT 1: Comparator filter output <= SDCMPL2.LLT Reset type: SYSRSn
2	IFH2	R	0h	High-level Interrupt flag for Ch2 0: Comparator filter output < SDCMPH2.HLT 1: Comparator filter output >= SDCMPH2.HLT Reset type: SYSRSn
1	IFL1	R	0h	Low-level Interrupt flag for Ch1 0: Comparator filter output > SDCMPL1.LLT 1: Comparator filter output <= SDCMPL1.LLT Reset type: SYSRSn
0	IFH1	R	0h	High-level Interrupt flag for Ch1 0: Comparator filter output < SDCMPH1.HLT 1: Comparator filter output >= SDCMPH1.HLT Reset type: SYSRSn

### 17.8.2.2 SDIFLGCLR Register (Offset = 2h) [reset = 0h]

SDIFLGCLR is shown in [Figure 17-13](#) and described in [Table 17-13](#).

Return to the [Summary Table](#).

SD Module Interrupt Flag Clear Bits:

Writing a "1" will clear the respective flag bit in the SDIFLG register.

Writes of "0" are ignored.

Note: If user writes a "1" to clear a bit on the same cycle that the hardware is trying to set the bit to "1", then hardware has priority and the bit will not be cleared.

**Figure 17-13. SDIFLGCLR Register**

31	30	29	28	27	26	25	24
MIF	RESERVED						
R=0/W=1-0h							
23	22	21	20	19	18	17	16
SDFINT4	SDFINT3	SDFINT2	SDFINT1	SDFOVF4	SDFOVF3	SDFOVF2	SDFOVF1
R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h
15	14	13	12	11	10	9	8
AF4	AF3	AF2	AF1	MF4	MF3	MF2	MF1
R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h
7	6	5	4	3	2	1	0
IFL4	IFH4	IFL3	IFH3	IFL2	IFH2	IFL1	IFH1
R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

**Table 17-13. SDIFLGCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MIF	R=0/W=1	0h	Flag-clear bit for SDFM Master Interrupt flag.  Writing a 1 to clear MIF flag in SDIFLG register  Writes of "0" are ignored.  Note: If the MIF flag is cleared and other Interrupts are still pending, MIF will again be set to 1 on the following SysClk cycle, and the INT output will be reasserted (pulsed low)  Reset type: SYSRSn
30-24	RESERVED	R=0	0h	Reserved
23	SDFINT4	R=0/W=1	0h	SDFIFO data ready Interrupt flag-clear bit for Ch4 Reset type: SYSRSn
22	SDFINT3	R=0/W=1	0h	SDFIFO data ready Interrupt flag-clear bit for Ch3 Reset type: SYSRSn
21	SDFINT2	R=0/W=1	0h	SDFIFO data ready Interrupt flag-clear bit for Ch2 Reset type: SYSRSn
20	SDFINT1	R=0/W=1	0h	SDFIFO data ready Interrupt flag-clear bit for Ch1 Reset type: SYSRSn
19	SDFOVF4	R=0/W=1	0h	SDFIFO overflow clear Ch4 Reset type: SYSRSn
18	SDFOVF3	R=0/W=1	0h	SDFIFO overflow clear Ch3 Reset type: SYSRSn
17	SDFOVF2	R=0/W=1	0h	SDFIFO overflow clear Ch2 Reset type: SYSRSn
16	SDFOVF1	R=0/W=1	0h	SDFIFO overflow clear Ch1 Reset type: SYSRSn

**Table 17-13. SDIFLGCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	AF4	R=0/W=1	0h	Flag-clear bit for Acknowledge flag for Filter 4 Reset type: SYSRSn
14	AF3	R=0/W=1	0h	Flag Clear bit for AF3 Reset type: SYSRSn
13	AF2	R=0/W=1	0h	Flag Clear bit for AF2 Reset type: SYSRSn
12	AF1	R=0/W=1	0h	Flag Clear bit for AF1 Reset type: SYSRSn
11	MF4	R=0/W=1	0h	Flag Clear bit for MF4 Reset type: SYSRSn
10	MF3	R=0/W=1	0h	Flag Clear bit for MF3 Reset type: SYSRSn
9	MF2	R=0/W=1	0h	Flag Clear bit for MF2 Reset type: SYSRSn
8	MF1	R=0/W=1	0h	Flag Clear bit for MF1 Reset type: SYSRSn
7	IFL4	R=0/W=1	0h	Flag Clear bit for IFL4 Reset type: SYSRSn
6	IFH4	R=0/W=1	0h	Flag Clear bit for IFH4 Reset type: SYSRSn
5	IFL3	R=0/W=1	0h	Flag Clear bit for IFL3 Reset type: SYSRSn
4	IFH3	R=0/W=1	0h	Flag Clear bit for IFH3 Reset type: SYSRSn
3	IFL2	R=0/W=1	0h	Flag Clear bit for IFL2 Reset type: SYSRSn
2	IFH2	R=0/W=1	0h	Flag Clear bit for IFH2 Reset type: SYSRSn
1	IFL1	R=0/W=1	0h	Flag Clear bit for IFL1 Reset type: SYSRSn
0	IFH1	R=0/W=1	0h	Flag Clear bit for IFH1 Reset type: SYSRSn

**17.8.2.3 SDCTL Register (Offset = 4h) [reset = 0h]**

SDCTL is shown in [Figure 17-14](#) and described in [Table 17-14](#).

Return to the [Summary Table](#).

SD Control Register

**Figure 17-14. SDCTL Register**

15	14	13	12	11	10	9	8
RESERVED	RESERVED	MIE	RESERVED				
R/W=0h							
7	6	5	4	3	2	1	0
RESERVED				HZ4	HZ3	HZ2	HZ1
				R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

**Table 17-14. SDCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14	RESERVED	R=0	0h	Reserved
13	MIE	R/W	0h	Master SDy_ERR interrupt enable 0: SDy_ERR Interrupt and interrupt flags are disabled 1: SDy_ERR Interrupt and interrupt flags are enabled Reset type: SYSRSn
12-4	RESERVED	R=0	0h	Reserved
3	HZ4	R=0/W=1	0h	Flag Clear bit for HZ4 Reset type: SYSRSn
2	HZ3	R=0/W=1	0h	Flag Clear bit for HZ3 Reset type: SYSRSn
1	HZ2	R=0/W=1	0h	Flag Clear bit for HZ2 Reset type: SYSRSn
0	HZ1	R=0/W=1	0h	Flag Clear bit for HZ1 Reset type: SYSRSn

**17.8.2.4 SDMFILEN Register (Offset = 6h) [reset = 0h]**

SDMFILEN is shown in [Figure 17-15](#) and described in [Table 17-15](#).

Return to the [Summary Table](#).

SD Master Filter Enable

**Figure 17-15. SDMFILEN Register**

15	14	13	12	11	10	9	8
RESERVED			RESERVED	MFE	RESERVED	RESERVED	RESERVED
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED	RESERVED			RESERVED			

**Table 17-15. SDMFILEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R=0	0h	Reserved
12	RESERVED	R=0	0h	Reserved
11	MFE	R/W	0h	Master Filter Enable  0: All the four data filter units of SDFM module are disabled. All FIFOs are cleared  1: Data filter units can be enabled if bit FEN is '1'. Reset type: SYSRSn
10	RESERVED	R=0	0h	Reserved
9	RESERVED	R=0	0h	Reserved
8-7	RESERVED	R=0	0h	Reserved
6-4	RESERVED	R=0	0h	Reserved
3-0	RESERVED	R=0	0h	Reserved



### 17.8.2.5 SDSTATUS Register (Offset = 7h) [reset = 0h]

SDSTATUS is shown in [Figure 17-16](#) and described in [Table 17-16](#).

Return to the [Summary Table](#).

SD Status Register

**Figure 17-16. SDSTATUS Register**

15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED				HZ4	HZ3	HZ2	HZ1
				R-0h	R-0h	R-0h	R-0h

**Table 17-16. SDSTATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	RESERVED	R	0h	Reserved
13	RESERVED	R	0h	Reserved
12	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7-4	RESERVED	R=0	0h	Reserved
3	HZ4	R	0h	High-level Threshold crossing (Z) flag Ch4  Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator IFHx flag, it does not have the ability to generate an interrupt.  0: Comparator filter output < SDCMPHZ4.HLTZ 1: Comparator filter output >= SDCMPHZ4.HLTZ Reset type: SYSRSn
2	HZ3	R	0h	High-level Threshold crossing (Z) flag Ch3  Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator IFHx flag, it does not have the ability to generate an interrupt.  0: Comparator filter output < SDCMPHZ3.HLTZ 1: Comparator filter output >= SDCMPHZ3.HLTZ Reset type: SYSRSn
1	HZ2	R	0h	High-level Threshold crossing (Z) flag Ch2  Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator IFHx flag, it does not have the ability to generate an interrupt.  0: Comparator filter output < SDCMPHZ2.HLTZ 1: Comparator filter output >= SDCMPHZ2.HLTZ Reset type: SYSRSn

**Table 17-16. SDSTATUS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	HZ1	R	0h	<p>High-level Threshold crossing (Z) flag Ch1</p> <p>Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator IFHx flag, it does not have the ability to generate an interrupt.</p> <p>0: Comparator filter output &lt; SDCMPHZ1.HLTZ  1: Comparator filter output &gt;= SDCMPHZ1.HLTZ</p> <p>Reset type: SYSRSn</p>

**17.8.2.6 SDCTLPARM1 Register (Offset = 10h) [reset = 0h]**

SDCTLPARM1 is shown in [Figure 17-17](#) and described in [Table 17-17](#).

Return to the [Summary Table](#).

Control Parameter Register for Ch1

**Figure 17-17. SDCTLPARM1 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			RESERVED	RESERVED	RESERVED	MOD	
R-0h						R/W-0h	

**Table 17-17. SDCTLPARM1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R=0	0h	Reserved
3	RESERVED	R=0	0h	Reserved
2	RESERVED	R=0	0h	Reserved
1-0	MOD	R/W	0h	Modulator clock modes MODE 0: Modulator clock running at 1x data rate MODE 1: modulator clock running at 1/2 data rate (dbl-edge clocking) MODE 2: modulator clock absent (Manchester encoded data) MODE 3: modulator clock running at 2x data rate Reset type: SYSRSn

**17.8.2.7 SDDFPARM1 Register (Offset = 11h) [reset = 0h]**

SDDFPARM1 is shown in [Figure 17-18](#) and described in [Table 17-18](#).

Return to the [Summary Table](#).

Data Filter Parameter Register for Ch1

**Figure 17-18. SDDFPARM1 Register**

15	14	13	12	11	10	9	8
RESERVED			SDSYNCEN	SST		AE	FEN
			R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DOSR							
R/W-0h							

**Table 17-18. SDDFPARM1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	SDSYNCEN	R/W	0h	PWM synchronization (SDSYNC) of data filter 0: PWM synchronization of data filter is disabled 1: PWM synchronization of data filter is enabled Note: SDSYNCS.SYNCSEL bits define which PWM signal is used to synchronize PWMs Reset type: SYSRSn
11-10	SST	R/W	0h	Data filter structure 00: Data filter runs with a Sincfast structure 01: Data filter runs with a Sinc1 structure 10: Data filter runs with a Sinc2 structure 11: Data filter runs with a Sinc3 structure Reset type: SYSRSn
9	AE	R/W	0h	Data filter Acknowledge Enable 0: Acknowledge flag is disabled for the particular filter 1: Acknowledge flag is enabled for the particular filter Reset type: SYSRSn
8	FEN	R/W	0h	Filter Enable 0: The data filter is disabled and no data is produced 1: The data filter is enabled and data are produced in the data filter Note: When filter is disabled, DOSR counter held in reset, filter data erased. Also resets FIFO pointers and clears the FIFO Reset type: SYSRSn
7-0	DOSR	R/W	0h	Data filter Oversampling ratio The actual oversampling ratio of data filter is DOSR + 1 These bits set the oversampling ratio of the data filter. 0x0FF represents an oversampling ratio of 256. Reset type: SYSRSn

**17.8.2.8 SDDPARM1 Register (Offset = 12h) [reset = 0h]**

SDDPARM1 is shown in [Figure 17-19](#) and described in [Table 17-19](#).

Return to the [Summary Table](#).

Data Parameter Register for Ch1

**Figure 17-19. SDDPARM1 Register**

15	14	13	12	11	10	9	8
		SH			DR	RESERVED	
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED							

**Table 17-19. SDDPARM1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	SH	R/W	0h	Shift Control These bits indicate by how many bits the 16-bit window is shifted up when 16-bit data representation is chosen. Reset type: SYSRSn
10	DR	R/W	0h	Data filter Data representation 0: Data stored in 16b 2's complement 1: Data stored in 32b 2's complement Reset type: SYSRSn
9-0	RESERVED	R=0	0h	Reserved

**17.8.2.9 SDCMPH1 Register (Offset = 13h) [reset = 7FFFh]**

SDCMPH1 is shown in [Figure 17-20](#) and described in [Table 17-20](#).

Return to the [Summary Table](#).

High-level Threshold Register for Ch1

**Figure 17-20. SDCMPH1 Register**

15	14	13	12	11	10	9	8
RESERVED	HLT						
R/W-7FFFh							
7	6	5	4	3	2	1	0
HLT							
R/W-7FFFh							

**Table 17-20. SDCMPH1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	HLT	R/W	7FFFh	Unsigned high-level threshold for the comparator filter output. Reset type: SYSRSn

**17.8.2.10 SDCMPL1 Register (Offset = 14h) [reset = 0h]**

SDCMPL1 is shown in [Figure 17-21](#) and described in [Table 17-21](#).

Return to the [Summary Table](#).

Low-level Threshold Register for Ch1

**Figure 17-21. SDCMPL1 Register**

15	14	13	12	11	10	9	8
RESERVED							LLT
R/W-0h							
7	6	5	4	3	2	1	0
LLT							
R/W-0h							

**Table 17-21. SDCMPL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	LLT	R/W	0h	Unsigned low-level threshold for the comparator filter output. Reset type: SYSRSn

**17.8.2.11 SDCPARAM1 Register (Offset = 15h) [reset = 0h]**

SDCPARM1 is shown in [Figure 17-22](#) and described in [Table 17-22](#).

Return to the [Summary Table](#).

Comparator Filter Parameter Register for Ch1

**Figure 17-22. SDCPARAM1 Register**

15		14		13		12		11		10		9		8	
RESERVED				CEN		RESERVED				HZEN		MFIE		CS1_CS0	
				R/W-0h								R/W-0h		R/W-0h	
7		6		5		4		3		2		1		0	
CS1_CS0		IEL		IEH						COSR					
R/W-0h		R/W-0h		R/W-0h						R/W-0h					

**Table 17-22. SDCPARAM1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R=0	0h	Reserved
13	CEN	R/W	0h	Comparator Filter enable 0: Disable comparator filter 1: Enable comparator filter Reset type: SYSRSn
12-11	RESERVED	R=0	0h	Reserved
10	HZEN	R/W	0h	High level (Z) Threshold crossing output enable 0: Disable Higher level Threshold (Z) crossing 1: Enable Higher level Threshold (Z) crossing Reset type: SYSRSn
9	MFIE	R/W	0h	Modulator Failure Interrupt Enable 0: Disable modulator failure interrupt and its flag 1: Enable modulator failure interrupt and its flag Reset type: SYSRSn
8-7	CS1_CS0	R/W	0h	Comparator filter structure 00: Comparator filter runs with a sincfast structure 01: Comparator filter runs with a Sinc1 structure 10: Comparator filter runs with a Sinc2 structure 11: Comparator filter runs with a Sinc3 structure Reset type: SYSRSn
6	IEL	R/W	0h	Low-level interrupt enable 0: Disable Lower Threshold interrupt 1: Enable Lower Threshold interrupt Reset type: SYSRSn
5	IEH	R/W	0h	High-level interrupt enable 0: Disable Higher Threshold interrupt 1: Enable Higher Threshold interrupt Reset type: SYSRSn



**Table 17-22. SDCPARM1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	COSR	R/W	0h	Comparator Oversampling ratio. The actual rate is COSR + 1. These bits set the oversampling ratio of the filter. 0x1F represents an oversampling ratio of 32 Reset type: SYSRSn

**17.8.2.12 SDDATA1 Register (Offset = 16h) [reset = 0h]**

SDDATA1 is shown in [Figure 17-23](#) and described in [Table 17-23](#).

Return to the [Summary Table](#).

Data Filter Data Register (16 or 32bit) for Ch1

**Figure 17-23. SDDATA1 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
DATA32HI																DATA16															
R-0h																R-0h															

**Table 17-23. SDDATA1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode, 16-bit Data in 16b mode Reset type: SYSRSn
15-0	DATA16	R	0h	Lo-order 16b in 32b mode Reset type: SYSRSn

**17.8.2.13 SDDATFIFO1 Register (Offset = 18h) [reset = 0h]**

SDDATFIFO1 is shown in [Figure 17-24](#) and described in [Table 17-24](#).

Return to the [Summary Table](#).

Filter Data FIFO Output(32b) for Ch1

**Figure 17-24. SDDATFIFO1 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
DATA32HI																DATA16															
R-0h																R-0h															

**Table 17-24. SDDATFIFO1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode, 16-bit Data in 16b mode Reset type: SYSRSn
15-0	DATA16	R	0h	Lo-order 16b in 32b mode Reset type: SYSRSn

**17.8.2.14 SDCDATA1 Register (Offset = 1Ah) [reset = 0h]**

SDCDATA1 is shown in [Figure 17-25](#) and described in [Table 17-25](#).

Return to the [Summary Table](#).

Comparator Filter Data Register (16b) for Ch1

**Figure 17-25. SDCDATA1 Register**

15	14	13	12	11	10	9	8
DATA16							
R-0h							
7	6	5	4	3	2	1	0
DATA16							
R-0h							

**Table 17-25. SDCDATA1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DATA16	R	0h	Comparator Data output - 16b only Reset type: SYSRSn

**17.8.2.15 SDCMPHZ1 Register (Offset = 1Ch) [reset = 0h]**

SDCMPHZ1 is shown in [Figure 17-26](#) and described in [Table 17-26](#).

Return to the [Summary Table](#).

High-level (Z) Threshold Register for Ch1

**Figure 17-26. SDCMPHZ1 Register**

15	14	13	12	11	10	9	8
RESERVED	HLTZ						
							R/W-0h
7	6	5	4	3	2	1	0
							HLTZ
							R/W-0h

**Table 17-26. SDCMPHZ1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	HLTZ	R/W	0h	Unsigned High-level threshold (Z) for the comparator filter output Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator SDCMPHx, it does not have the ability to generate an interrupt. Reset type: SYSRSn

**17.8.2.16 SDFIFOCTL1 Register (Offset = 1Dh) [reset = 0h]**

SDFIFOCTL1 is shown in [Figure 17-27](#) and described in [Table 17-27](#).

Return to the [Summary Table](#).

FIFO Control Register for Ch1

**Figure 17-27. SDFIFOCTL1 Register**

15		14		13		12		11		10		9		8	
OVFIEN		DRINTSEL		FFEN		FFIEN		RESERVED				SDFFST			
R/W-0h		R/W-0h		R/W-0h		R/W-0h						R-0h			
7		6		5		4		3		2		1		0	
		SDFFST		RESERVED						SDFFIL					
		R-0h								R/W-0h					

**Table 17-27. SDFIFOCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	OVFIEN	R/W	0h	SDFIFO Overflow interrupt enable 0: SDFIFO Overflow condition will not generate an interrupt 1: SDFIFO overflow condition generates an interrupt on SDy_ERR Reset type: SYSRSn
14	DRINTSEL	R/W	0h	Data-Ready Interrupt (DRINT) source select 0 = AF1 (Select non-FIFO data-ready interrupt) 1 = SDFINT1 (Select FIFO data-ready interrupt) Reset type: SYSRSn
13	FFEN	R/W	0h	SDFIFO Enable 0: Disable FIFO operation 1: Enable FIFO operation Note: When FIFO is disabled, FIFO contents are cleared Reset type: SYSRSn
12	FFIEN	R/W	0h	SDFIFO data ready Interrupt Enable Reset type: SYSRSn
11	RESERVED	R=0	0h	Reserved
10-6	SDFFST	R	0h	SDFIFO Status 00000 FIFO empty 00001 FIFO has 1 word ... 10000 FIFO has 16 words Reset type: SYSRSn
5	RESERVED	R=0	0h	Reserved
4-0	SDFFIL	R/W	0h	SDFIFO interrupt level bits The FIFO will generate an interrupt when the FIFO status (SDFFST) >= FIFO level (SDFFIL) Reset type: SYSRSn

**17.8.2.17 SDSYNC1 Register (Offset = 1Eh) [reset = 400h]**

SDSYNC1 is shown in [Figure 17-28](#) and described in [Table 17-28](#).

Return to the [Summary Table](#).

SD Filter Sync control for Ch1

**Figure 17-28. SDSYNC1 Register**

15	14	13	12	11	10	9	8
RESERVED					WTSCLEN	FFSYNCLREN	WTSYNCLR
					R/W-1h	R/W-0h	R=0/W-0h
7	6	5	4	3	2	1	0
WTSYNFLG	WTSYNEN	SYNSEL					
R-0h	R/W-0h	R/W-0h					

**Table 17-28. SDSYNC1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R=0	0h	Reserved
10	WTSCLEN	R/W	1h	WTSYNFLG Clear-on-FIFOINT Enable 0: WTSYNFLG can only be cleared manually (using WTSYNCLR bit) 1: WTSYNFLG is cleared automatically on SDFINT Reset type: SYSRSn
9	FFSYNCLREN	R/W	0h	FIFO Clear-on-SDSYNC Enable 0: SDFIFO is not automatically cleared upon receiving SDSYNC 1: SDFIFO is automatically cleared upon receiving SDSYNC Reset type: SYSRSn
8	WTSYNCLR	R=0/W	0h	Wait-for-Sync Flag Clear (always reads 0) 0: Write of 0 has no affect 1: Write of 1 clears WTSYNFLG Reset type: SYSRSn
7	WTSYNFLG	R	0h	Wait-for-Sync Flag 0: SDSYNC event has not occurred 1: SDSYNC event occurred. Reset type: SYSRSn
6	WTSYNEN	R/W	0h	Wait-for-Sync Enable 0: Incoming Data written to SDFIFO on every Data-Ready (DR) Event 1: Incoming Data written to SDFIFO on DR event only after SDSYNC event occurs Reset type: SYSRSn
5-0	SYNSEL	R/W	0h	Defines source for the SDSYNC Input on this channel Refer SDSYNcx.SYNSEL table Reset type: SYSRSn

**17.8.2.18 SDCTLPARM2 Register (Offset = 20h) [reset = 0h]**

SDCTLPARM2 is shown in [Figure 17-29](#) and described in [Table 17-29](#).

Return to the [Summary Table](#).

Control Parameter Register for Ch2

**Figure 17-29. SDCTLPARM2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			RESERVED	RESERVED	RESERVED	MOD	
R-0h						R/W-0h	

**Table 17-29. SDCTLPARM2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R=0	0h	Reserved
3	RESERVED	R=0	0h	Reserved
2	RESERVED	R=0	0h	Reserved
1-0	MOD	R/W	0h	Modulator clock modes MODE 0: Modulator clock running at 1x data rate MODE 1: modulator clock running at 1/2 data rate (dbl-edge clocking) MODE 2: modulator clock absent (Manchester encoded data) MODE 3: modulator clock running at 2x data rate Reset type: SYSRSn



**17.8.2.19 SDDFPARM2 Register (Offset = 21h) [reset = 0h]**

SDDFPARM2 is shown in [Figure 17-30](#) and described in [Table 17-30](#).

Return to the [Summary Table](#).

Data Filter Parameter Register for Ch2

**Figure 17-30. SDDFPARM2 Register**

15	14	13	12	11	10	9	8
RESERVED			SDSYNCEN	SST		AE	FEN
			R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DOSR							
R/W-0h							

**Table 17-30. SDDFPARM2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	SDSYNCEN	R/W	0h	PWM synchronization (SDSYNC) of data filter 0: PWM synchronization of data filter is disabled 1: PWM synchronization of data filter is enabled Note: SDSYNCS.SYNCSEL bits define which PWM signal is used to synchronize PWMs Reset type: SYSRSn
11-10	SST	R/W	0h	Data filter structure 00: Data filter runs with a Sincfast structure 01: Data filter runs with a Sinc1 structure 10: Data filter runs with a Sinc2 structure 11: Data filter runs with a Sinc3 structure Reset type: SYSRSn
9	AE	R/W	0h	Data filter Acknowledge Enable 0: Acknowledge flag is disabled for the particular filter 1: Acknowledge flag is enabled for the particular filter Reset type: SYSRSn
8	FEN	R/W	0h	Filter Enable 0: The data filter is disabled and no data is produced 1: The data filter is enabled and data are produced in the data filter Note: When filter is disabled, DOSR counter held in reset, filter data erased. Also resets FIFO pointers and clears the FIFO Reset type: SYSRSn
7-0	DOSR	R/W	0h	Data filter Oversampling ratio The actual oversampling ratio of data filter is DOSR + 1 These bits set the oversampling ratio of the data filter. 0x0FF represents an oversampling ratio of 256. Reset type: SYSRSn

**17.8.2.20 SDDPARM2 Register (Offset = 22h) [reset = 0h]**

SDDPARM2 is shown in [Figure 17-31](#) and described in [Table 17-31](#).

Return to the [Summary Table](#).

Data Parameter Register for Ch2

**Figure 17-31. SDDPARM2 Register**

15	14	13	12	11	10	9	8
SH					DR	RESERVED	
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED							

**Table 17-31. SDDPARM2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	SH	R/W	0h	Shift Control These bits indicate by how many bits the 16-bit window is shifted up when 16-bit data representation is chosen. Reset type: SYSRSn
10	DR	R/W	0h	Data filter Data representation 0: Data stored in 16b 2's complement 1: Data stored in 32b 2's complement Reset type: SYSRSn
9-0	RESERVED	R=0	0h	Reserved

**17.8.2.21 SDCMPH2 Register (Offset = 23h) [reset = 7FFFh]**

SDCMPH2 is shown in [Figure 17-32](#) and described in [Table 17-32](#).

Return to the [Summary Table](#).

High-level Threshold Register for Ch2

**Figure 17-32. SDCMPH2 Register**

15	14	13	12	11	10	9	8
RESERVED	HLT						
R/W-7FFFh							
7	6	5	4	3	2	1	0
HLT							
R/W-7FFFh							

**Table 17-32. SDCMPH2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	HLT	R/W	7FFFh	Unsigned high-level threshold for the comparator filter output. Reset type: SYSRSn

**17.8.2.22 SDCMPL2 Register (Offset = 24h) [reset = 0h]**

SDCMPL2 is shown in [Figure 17-33](#) and described in [Table 17-33](#).

Return to the [Summary Table](#).

Low-level Threshold Register for Ch2

**Figure 17-33. SDCMPL2 Register**

15	14	13	12	11	10	9	8
RESERVED							LLT
R/W-0h							
7	6	5	4	3	2	1	0
LLT							
R/W-0h							

**Table 17-33. SDCMPL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	LLT	R/W	0h	Unsigned low-level threshold for the comparator filter output. Reset type: SYSRSn

### 17.8.2.23 SDCPARAM2 Register (Offset = 25h) [reset = 0h]

SDCPARM2 is shown in [Figure 17-34](#) and described in [Table 17-34](#).

Return to the [Summary Table](#).

Comparator Filter Parameter Register for Ch2

**Figure 17-34. SDCPARAM2 Register**

15	14	13	12	11	10	9	8	
RESERVED		CEN	RESERVED		HZEN	MFIE	CS1_CS0	
		R/W-0h			R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0	
CS1_CS0	IEL	IEH	COSR					
R/W-0h	R/W-0h	R/W-0h	R/W-0h					

**Table 17-34. SDCPARAM2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R=0	0h	Reserved
13	CEN	R/W	0h	Comparator Filter enable 0: Disable comparator filter 1: Enable comparator filter Reset type: SYSRSn
12-11	RESERVED	R=0	0h	Reserved
10	HZEN	R/W	0h	High level (Z) Threshold crossing output enable 0: Disable Higher level Threshold (Z) crossing 1: Enable Higher level Threshold (Z) crossing Reset type: SYSRSn
9	MFIE	R/W	0h	Modulator Failure Interrupt Enable 0: Disable modulator failure interrupt and its flag 1: Enable modulator failure interrupt and its flag Reset type: SYSRSn
8-7	CS1_CS0	R/W	0h	Comparator filter structure 00: Comparator filter runs with a sincfast structure 01: Comparator filter runs with a Sinc1 structure 10: Comparator filter runs with a Sinc2 structure 11: Comparator filter runs with a Sinc3 structure Reset type: SYSRSn
6	IEL	R/W	0h	Low-level interrupt enable 0: Disable Lower Threshold interrupt 1: Enable Lower Threshold interrupt Reset type: SYSRSn
5	IEH	R/W	0h	High-level interrupt enable 0: Disable Higher Threshold interrupt 1: Enable Higher Threshold interrupt Reset type: SYSRSn

**Table 17-34. SDCPARM2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	COSR	R/W	0h	Comparator Oversampling ratio. The actual rate is COSR + 1. These bits set the oversampling ratio of the filter. 0x1F represents an oversampling ratio of 32 Reset type: SYSRSn

**17.8.2.24 SDDATA2 Register (Offset = 26h) [reset = 0h]**

SDDATA2 is shown in [Figure 17-35](#) and described in [Table 17-35](#).

Return to the [Summary Table](#).

Data Filter Data Register (16 or 32bit) for Ch2

**Figure 17-35. SDDATA2 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
DATA32HI																DATA16															
R-0h																R-0h															

**Table 17-35. SDDATA2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode, 16-bit Data in 16b mode Reset type: SYSRSn
15-0	DATA16	R	0h	Lo-order 16b in 32b mode Reset type: SYSRSn

**17.8.2.25 SDDATFIFO2 Register (Offset = 28h) [reset = 0h]**

SDDATFIFO2 is shown in [Figure 17-36](#) and described in [Table 17-36](#).

Return to the [Summary Table](#).

Filter Data FIFO Output(32b) for Ch2

**Figure 17-36. SDDATFIFO2 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
DATA32HI																DATA16															
R-0h																R-0h															

**Table 17-36. SDDATFIFO2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode, 16-bit Data in 16b mode Reset type: SYSRSn
15-0	DATA16	R	0h	Lo-order 16b in 32b mode Reset type: SYSRSn



**17.8.2.26 SDCDATA2 Register (Offset = 2Ah) [reset = 0h]**

SDCDATA2 is shown in [Figure 17-37](#) and described in [Table 17-37](#).

Return to the [Summary Table](#).

Comparator Filter Data Register (16b) for Ch2

**Figure 17-37. SDCDATA2 Register**

15	14	13	12	11	10	9	8
DATA16							
R-0h							
7	6	5	4	3	2	1	0
DATA16							
R-0h							

**Table 17-37. SDCDATA2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DATA16	R	0h	Comparator Data output - 16b only Reset type: SYSRSn

**17.8.2.27 SDCMPHZ2 Register (Offset = 2Ch) [reset = 0h]**

SDCMPHZ2 is shown in [Figure 17-38](#) and described in [Table 17-38](#).

Return to the [Summary Table](#).

High-level (Z) Threshold Register for Ch2

**Figure 17-38. SDCMPHZ2 Register**

15	14	13	12	11	10	9	8
RESERVED	HLTZ						
R/W-0h							
7	6	5	4	3	2	1	0
HLTZ							
R/W-0h							

**Table 17-38. SDCMPHZ2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	HLTZ	R/W	0h	Unsigned High-level threshold (Z) for the comparator filter output Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator SDCMPHx, it does not have the ability to generate an interrupt. Reset type: SYSRSn

### 17.8.2.28 SDFIFOCTL2 Register (Offset = 2Dh) [reset = 0h]

SDFIFOCTL2 is shown in [Figure 17-39](#) and described in [Table 17-39](#).

Return to the [Summary Table](#).

FIFO Control Register for Ch2

**Figure 17-39. SDFIFOCTL2 Register**

15		14		13		12		11		10		9		8	
OVFIEN		DRINTSEL		FFEN		FFIEN		RESERVED				SDFFST			
R/W-0h		R/W-0h		R/W-0h		R/W-0h						R-0h			
7		6		5		4		3		2		1		0	
		SDFFST		RESERVED						SDFFIL					
		R-0h								R/W-0h					

**Table 17-39. SDFIFOCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	OVFIEN	R/W	0h	SDFIFO Overflow interrupt enable 0: SDFIFO Overflow condition will not generate an interrupt 1: SDFIFO overflow condition generates an interrupt on SDy_ERR Reset type: SYSRSn
14	DRINTSEL	R/W	0h	Data-Ready Interrupt (DRINT) source select 0 = AF1 (Select non-FIFO data-ready interrupt) 1 = SDFINT1 (Select FIFO data-ready interrupt) Reset type: SYSRSn
13	FFEN	R/W	0h	SDFIFO Enable 0: Disable FIFO operation 1: Enable FIFO operation Note: When FIFO is disabled, FIFO contents are cleared Reset type: SYSRSn
12	FFIEN	R/W	0h	SDFIFO data ready Interrupt Enable Reset type: SYSRSn
11	RESERVED	R=0	0h	Reserved
10-6	SDFFST	R	0h	SDFIFO Status 00000 FIFO empty 00001 FIFO has 1 word ... 10000 FIFO has 16 words Reset type: SYSRSn
5	RESERVED	R=0	0h	Reserved
4-0	SDFFIL	R/W	0h	SDFIFO interrupt level bits The FIFO will generate an interrupt when the FIFO status (SDFFST) >= FIFO level (SDFFIL) Reset type: SYSRSn

**17.8.2.29 SDSYNC2 Register (Offset = 2Eh) [reset = 400h]**

SDSYNC2 is shown in [Figure 17-40](#) and described in [Table 17-40](#).

Return to the [Summary Table](#).

SD Filter Sync control for Ch2

**Figure 17-40. SDSYNC2 Register**

15	14	13	12	11	10	9	8	
RESERVED					WTSCLREN	FFSYNCCLE N	WTSYNCLR	
					R/W-1h	R/W-0h	R=0/W-0h	
7	6	5	4	3	2	1	0	
WTSYNFLG	WTSYNCEN	SYNCSEL						
R-0h	R/W-0h	R/W-0h						

**Table 17-40. SDSYNC2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R=0	0h	Reserved
10	WTSCLREN	R/W	1h	WTSYNFLG Clear-on-FIFOINT Enable 0: WTSYNFLG can only be cleared manually (using WTSYNCLR bit) 1: WTSYNFLG is cleared automatically on SDFINT Reset type: SYSRSn
9	FFSYNCCLEN	R/W	0h	FIFO Clear-on-SDSYNC Enable 0: SDFIFO is not automatically cleared upon receiving SDSYNC 1: SDFIFO is automatically cleared upon receiving SDSYNC Reset type: SYSRSn
8	WTSYNCLR	R=0/W	0h	Wait-for-Sync Flag Clear (always reads 0) 0: Write of 0 has no affect 1: Write of 1 clears WTSYNFLG Reset type: SYSRSn
7	WTSYNFLG	R	0h	Wait-for-Sync Flag 0: SDSYNC event has not occurred 1: SDSYNC event occurred. Reset type: SYSRSn
6	WTSYNCEN	R/W	0h	Wait-for-Sync Enable 0: Incoming Data written to SDFIFO on every Data-Ready (DR) Event 1: Incoming Data written to SDFIFO on DR event only after SDSYNC event occurs Reset type: SYSRSn
5-0	SYNCSEL	R/W	0h	Defines source for the SDSYNC Input on this channel Refer SDSYNcx.SYNCSEL table Reset type: SYSRSn

**17.8.2.30 SDCTLPARM3 Register (Offset = 30h) [reset = 0h]**

SDCTLPARM3 is shown in [Figure 17-41](#) and described in [Table 17-41](#).

Return to the [Summary Table](#).

Control Parameter Register for Ch3

**Figure 17-41. SDCTLPARM3 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			RESERVED	RESERVED	RESERVED	MOD	
R-0h						R/W-0h	

**Table 17-41. SDCTLPARM3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R=0	0h	Reserved
3	RESERVED	R=0	0h	Reserved
2	RESERVED	R=0	0h	Reserved
1-0	MOD	R/W	0h	Modulator clock modes MODE 0: Modulator clock running at 1x data rate MODE 1: modulator clock running at 1/2 data rate (dbl-edge clocking) MODE 2: modulator clock absent (Manchester encoded data) MODE 3: modulator clock running at 2x data rate Reset type: SYSRSn

**17.8.2.31 SDDFPARM3 Register (Offset = 31h) [reset = 0h]**

SDDFPARM3 is shown in [Figure 17-42](#) and described in [Table 17-42](#).

Return to the [Summary Table](#).

Data Filter Parameter Register for Ch3

**Figure 17-42. SDDFPARM3 Register**

15	14	13	12	11	10	9	8
RESERVED			SDSYNCEN	SST		AE	FEN
			R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DOSR							
R/W-0h							

**Table 17-42. SDDFPARM3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	SDSYNCEN	R/W	0h	PWM synchronization (SDSYNC) of data filter 0: PWM synchronization of data filter is disabled 1: PWM synchronization of data filter is enabled Note: SDSYNCS.SYNCSEL bits define which PWM signal is used to synchronize PWMs Reset type: SYSRSn
11-10	SST	R/W	0h	Data filter structure 00: Data filter runs with a Sincfast structure 01: Data filter runs with a Sinc1 structure 10: Data filter runs with a Sinc2 structure 11: Data filter runs with a Sinc3 structure Reset type: SYSRSn
9	AE	R/W	0h	Data filter Acknowledge Enable 0: Acknowledge flag is disabled for the particular filter 1: Acknowledge flag is enabled for the particular filter Reset type: SYSRSn
8	FEN	R/W	0h	Filter Enable 0: The data filter is disabled and no data is produced 1: The data filter is enabled and data are produced in the data filter Note: When filter is disabled, DOSR counter held in reset, filter data erased. Also resets FIFO pointers and clears the FIFO Reset type: SYSRSn
7-0	DOSR	R/W	0h	Data filter Oversampling ratio The actual oversampling ratio of data filter is DOSR + 1 These bits set the oversampling ratio of the data filter. 0x0FF represents an oversampling ratio of 256. Reset type: SYSRSn

**17.8.2.32 SDDPARM3 Register (Offset = 32h) [reset = 0h]**

SDDPARM3 is shown in [Figure 17-43](#) and described in [Table 17-43](#).

Return to the [Summary Table](#).

Data Parameter Register for Ch3

**Figure 17-43. SDDPARM3 Register**

15	14	13	12	11	10	9	8
SH				DR		RESERVED	
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED							

**Table 17-43. SDDPARM3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	SH	R/W	0h	Shift Control These bits indicate by how many bits the 16-bit window is shifted up when 16-bit data representation is chosen. Reset type: SYSRSn
10	DR	R/W	0h	Data filter Data representation 0: Data stored in 16b 2's complement 1: Data stored in 32b 2's complement Reset type: SYSRSn
9-0	RESERVED	R=0	0h	Reserved

**17.8.2.33 SDCMPH3 Register (Offset = 33h) [reset = 7FFFh]**

SDCMPH3 is shown in [Figure 17-44](#) and described in [Table 17-44](#).

Return to the [Summary Table](#).

High-level Threshold Register for Ch3

**Figure 17-44. SDCMPH3 Register**

15	14	13	12	11	10	9	8
RESERVED	HLT						
R/W-7FFFh							
7	6	5	4	3	2	1	0
HLT							
R/W-7FFFh							

**Table 17-44. SDCMPH3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	HLT	R/W	7FFFh	Unsigned high-level threshold for the comparator filter output. Reset type: SYSRSn



**17.8.2.34 SDCMPL3 Register (Offset = 34h) [reset = 0h]**

SDCMPL3 is shown in [Figure 17-45](#) and described in [Table 17-45](#).

Return to the [Summary Table](#).

Low-level Threshold Register for Ch3

**Figure 17-45. SDCMPL3 Register**

15	14	13	12	11	10	9	8
RESERVED							LLT
R/W-0h							
7	6	5	4	3	2	1	0
LLT							
R/W-0h							

**Table 17-45. SDCMPL3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	LLT	R/W	0h	Unsigned low-level threshold for the comparator filter output. Reset type: SYSRSn

**17.8.2.35 SDCPARM3 Register (Offset = 35h) [reset = 0h]**

SDCPARM3 is shown in [Figure 17-46](#) and described in [Table 17-46](#).

Return to the [Summary Table](#).

Comparator Filter Parameter Register for Ch3

**Figure 17-46. SDCPARM3 Register**

15	14	13	12	11	10	9	8	
RESERVED		CEN	RESERVED		HZEN	MFIE	CS1_CS0	
		R/W-0h			R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0	
CS1_CS0	IEL	IEH	COSR					
R/W-0h	R/W-0h	R/W-0h	R/W-0h					

**Table 17-46. SDCPARM3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R=0	0h	Reserved
13	CEN	R/W	0h	Comparator Filter enable 0: Disable comparator filter 1: Enable comparator filter Reset type: SYSRSn
12-11	RESERVED	R=0	0h	Reserved
10	HZEN	R/W	0h	High level (Z) Threshold crossing output enable 0: Disable Higher level Threshold (Z) crossing 1: Enable Higher level Threshold (Z) crossing Reset type: SYSRSn
9	MFIE	R/W	0h	Modulator Failure Interrupt Enable 0: Disable modulator failure interrupt and its flag 1: Enable modulator failure interrupt and its flag Reset type: SYSRSn
8-7	CS1_CS0	R/W	0h	Comparator filter structure 00: Comparator filter runs with a sincfast structure 01: Comparator filter runs with a Sinc1 structure 10: Comparator filter runs with a Sinc2 structure 11: Comparator filter runs with a Sinc3 structure Reset type: SYSRSn
6	IEL	R/W	0h	Low-level interrupt enable 0: Disable Lower Threshold interrupt 1: Enable Lower Threshold interrupt Reset type: SYSRSn
5	IEH	R/W	0h	High-level interrupt enable 0: Disable Higher Threshold interrupt 1: Enable Higher Threshold interrupt Reset type: SYSRSn

**Table 17-46. SDCPARM3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	COSR	R/W	0h	<p>Comparator Oversampling ratio.</p> <p>The actual rate is COSR + 1.</p> <p>These bits set the oversampling ratio of the filter.</p> <p>0x1F represents an oversampling ratio of 32</p> <p>Reset type: SYSRSn</p>

**17.8.2.36 SDDATA3 Register (Offset = 36h) [reset = 0h]**

SDDATA3 is shown in [Figure 17-47](#) and described in [Table 17-47](#).

Return to the [Summary Table](#).

Data Filter Data Register (16 or 32bit) for Ch3

**Figure 17-47. SDDATA3 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
DATA32HI																DATA16															
R-0h																R-0h															

**Table 17-47. SDDATA3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode, 16-bit Data in 16b mode Reset type: SYSRSn
15-0	DATA16	R	0h	Lo-order 16b in 32b mode Reset type: SYSRSn

**17.8.2.37 SDDATFIFO3 Register (Offset = 38h) [reset = 0h]**

SDDATFIFO3 is shown in [Figure 17-48](#) and described in [Table 17-48](#).

Return to the [Summary Table](#).

Filter Data FIFO Output(32b) for Ch3

**Figure 17-48. SDDATFIFO3 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
DATA32HI																DATA16															
R-0h																R-0h															

**Table 17-48. SDDATFIFO3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode, 16-bit Data in 16b mode Reset type: SYSRSn
15-0	DATA16	R	0h	Lo-order 16b in 32b mode Reset type: SYSRSn

**17.8.2.38 SDCDATA3 Register (Offset = 3Ah) [reset = 0h]**

SDCDATA3 is shown in [Figure 17-49](#) and described in [Table 17-49](#).

Return to the [Summary Table](#).

Comparator Filter Data Register (16b) for Ch3

**Figure 17-49. SDCDATA3 Register**

15	14	13	12	11	10	9	8
DATA16							
R-0h							
7	6	5	4	3	2	1	0
DATA16							
R-0h							

**Table 17-49. SDCDATA3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DATA16	R	0h	Comparator Data output - 16b only Reset type: SYSRSn

**17.8.2.39 SDCMPHZ3 Register (Offset = 3Ch) [reset = 0h]**

SDCMPHZ3 is shown in [Figure 17-50](#) and described in [Table 17-50](#).

Return to the [Summary Table](#).

High-level (Z) Threshold Register for Ch3

**Figure 17-50. SDCMPHZ3 Register**

15	14	13	12	11	10	9	8
RESERVED	HLTZ						
							R/W-0h
7	6	5	4	3	2	1	0
							HLTZ
							R/W-0h

**Table 17-50. SDCMPHZ3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	HLTZ	R/W	0h	Unsigned High-level threshold (Z) for the comparator filter output Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator SDCMPHx, it does not have the ability to generate an interrupt. Reset type: SYSRSn

**17.8.2.40 SDFIFOCTL3 Register (Offset = 3Dh) [reset = 0h]**

 SDFIFOCTL3 is shown in [Figure 17-51](#) and described in [Table 17-51](#).

 Return to the [Summary Table](#).

FIFO Control Register for Ch3

**Figure 17-51. SDFIFOCTL3 Register**

15		14		13		12		11		10		9		8	
OVFIEN		DRINTSEL		FFEN		FFIEN		RESERVED				SDFFST			
R/W-0h		R/W-0h		R/W-0h		R/W-0h						R-0h			
7		6		5		4		3		2		1		0	
SDFFST				RESERVED						SDFFIL					
R-0h										R/W-0h					

**Table 17-51. SDFIFOCTL3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	OVFIEN	R/W	0h	SDFIFO Overflow interrupt enable 0: SDFIFO Overflow condition will not generate an interrupt 1: SDFIFO overflow condition generates an interrupt on SDy_ERR Reset type: SYSRSn
14	DRINTSEL	R/W	0h	Data-Ready Interrupt (DRINT) source select 0 = AF1 (Select non-FIFO data-ready interrupt) 1 = SDFINT1 (Select FIFO data-ready interrupt) Reset type: SYSRSn
13	FFEN	R/W	0h	SDFIFO Enable 0: Disable FIFO operation 1: Enable FIFO operation Note: When FIFO is disabled, FIFO contents are cleared Reset type: SYSRSn
12	FFIEN	R/W	0h	SDFIFO data ready Interrupt Enable Reset type: SYSRSn
11	RESERVED	R=0	0h	Reserved
10-6	SDFFST	R	0h	SDFIFO Status 00000 FIFO empty 00001 FIFO has 1 word ... 10000 FIFO has 16 words Reset type: SYSRSn
5	RESERVED	R=0	0h	Reserved
4-0	SDFFIL	R/W	0h	SDFIFO interrupt level bits The FIFO will generate an interrupt when the FIFO status (SDFFST) >= FIFO level (SDFFIL) Reset type: SYSRSn



**17.8.2.41 SDSYNC3 Register (Offset = 3Eh) [reset = 400h]**

SDSYNC3 is shown in [Figure 17-52](#) and described in [Table 17-52](#).

Return to the [Summary Table](#).

SD Filter Sync control for Ch3

**Figure 17-52. SDSYNC3 Register**

15	14	13	12	11	10	9	8
RESERVED					WTSCLEN	FFSYNCLREN	WTSYNCLR
					R/W-1h	R/W-0h	R=0/W-0h
7	6	5	4	3	2	1	0
WTSYNFLG	WTSYNEN	SYNCSEL					
R-0h	R/W-0h	R/W-0h					

**Table 17-52. SDSYNC3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R=0	0h	Reserved
10	WTSCLEN	R/W	1h	WTSYNFLG Clear-on-FIFOINT Enable 0: WTSYNFLG can only be cleared manually (using WTSYNCLR bit) 1: WTSYNFLG is cleared automatically on SDFINT Reset type: SYSRSn
9	FFSYNCLREN	R/W	0h	FIFO Clear-on-SDSYNC Enable 0: SDFIFO is not automatically cleared upon receiving SDSYNC 1: SDFIFO is automatically cleared upon receiving SDSYNC Reset type: SYSRSn
8	WTSYNCLR	R=0/W	0h	Wait-for-Sync Flag Clear (always reads 0) 0: Write of 0 has no affect 1: Write of 1 clears WTSYNFLG Reset type: SYSRSn
7	WTSYNFLG	R	0h	Wait-for-Sync Flag 0: SDSYNC event has not occurred 1: SDSYNC event occurred. Reset type: SYSRSn
6	WTSYNEN	R/W	0h	Wait-for-Sync Enable 0: Incoming Data written to SDFIFO on every Data-Ready (DR) Event 1: Incoming Data written to SDFIFO on DR event only after SDSYNC event occurs Reset type: SYSRSn
5-0	SYNCSEL	R/W	0h	Defines source for the SDSYNC Input on this channel Refer SDSYNcx.SYNCSEL table Reset type: SYSRSn

**17.8.2.42 SDCTLPARM4 Register (Offset = 40h) [reset = 0h]**

SDCTLPARM4 is shown in [Figure 17-53](#) and described in [Table 17-53](#).

Return to the [Summary Table](#).

Control Parameter Register for Ch4

**Figure 17-53. SDCTLPARM4 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			RESERVED	RESERVED	RESERVED	MOD	
R-0h						R/W-0h	

**Table 17-53. SDCTLPARM4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R=0	0h	Reserved
3	RESERVED	R=0	0h	Reserved
2	RESERVED	R=0	0h	Reserved
1-0	MOD	R/W	0h	Modulator clock modes MODE 0: Modulator clock running at 1x data rate MODE 1: modulator clock running at 1/2 data rate (dbl-edge clocking) MODE 2: modulator clock absent (Manchester encoded data) MODE 3: modulator clock running at 2x data rate Reset type: SYSRSn

### 17.8.2.43 SDDFPARM4 Register (Offset = 41h) [reset = 0h]

SDDFPARM4 is shown in [Figure 17-54](#) and described in [Table 17-54](#).

Return to the [Summary Table](#).

Data Filter Parameter Register for Ch4

**Figure 17-54. SDDFPARM4 Register**

15	14	13	12	11	10	9	8
RESERVED			SDSYNCEN	SST		AE	FEN
			R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DOSR							
R/W-0h							

**Table 17-54. SDDFPARM4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	SDSYNCEN	R/W	0h	PWM synchronization (SDSYNC) of data filter 0: PWM synchronization of data filter is disabled 1: PWM synchronization of data filter is enabled Note: SDSYNcx.SYNCSEL bits define which PWM signal is used to synchronize PWMs Reset type: SYSRSn
11-10	SST	R/W	0h	Data filter structure 00: Data filter runs with a Sincfast structure 01: Data filter runs with a Sinc1 structure 10: Data filter runs with a Sinc2 structure 11: Data filter runs with a Sinc3 structure Reset type: SYSRSn
9	AE	R/W	0h	Data filter Acknowledge Enable 0: Acknowledge flag is disabled for the particular filter 1: Acknowledge flag is enabled for the particular filter Reset type: SYSRSn
8	FEN	R/W	0h	Filter Enable 0: The data filter is disabled and no data is produced 1: The data filter is enabled and data are produced in the data filter Note: When filter is disabled, DOSR counter held in reset, filter data erased. Also resets FIFO pointers and clears the FIFO Reset type: SYSRSn
7-0	DOSR	R/W	0h	Data filter Oversampling ratio The actual oversampling ratio of data filter is DOSR + 1 These bits set the oversampling ratio of the data filter. 0x0FF represents an oversampling ratio of 256. Reset type: SYSRSn

**17.8.2.44 SDDPARM4 Register (Offset = 42h) [reset = 0h]**

SDDPARM4 is shown in [Figure 17-55](#) and described in [Table 17-55](#).

Return to the [Summary Table](#).

Data Parameter Register for Ch4

**Figure 17-55. SDDPARM4 Register**

15	14	13	12	11	10	9	8
SH					DR	RESERVED	
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED							

**Table 17-55. SDDPARM4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	SH	R/W	0h	Shift Control These bits indicate by how many bits the 16-bit window is shifted up when 16-bit data representation is chosen. Reset type: SYSRSn
10	DR	R/W	0h	Data filter Data representation 0: Data stored in 16b 2's complement 1: Data stored in 32b 2's complement Reset type: SYSRSn
9-0	RESERVED	R=0	0h	Reserved

**17.8.2.45 SDCMPH4 Register (Offset = 43h) [reset = 7FFFh]**

SDCMPH4 is shown in [Figure 17-56](#) and described in [Table 17-56](#).

Return to the [Summary Table](#).

High-level Threshold Register for Ch4

**Figure 17-56. SDCMPH4 Register**

15	14	13	12	11	10	9	8
RESERVED	HLT						
R/W-7FFFh							
7	6	5	4	3	2	1	0
HLT							
R/W-7FFFh							

**Table 17-56. SDCMPH4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	HLT	R/W	7FFFh	Unsigned high-level threshold for the comparator filter output. Reset type: SYSRSn

**17.8.2.46 SDCMPL4 Register (Offset = 44h) [reset = 0h]**

SDCMPL4 is shown in [Figure 17-57](#) and described in [Table 17-57](#).

Return to the [Summary Table](#).

Low-level Threshold Register for Ch4

**Figure 17-57. SDCMPL4 Register**

15	14	13	12	11	10	9	8
RESERVED	LLT						
R/W-0h							
7	6	5	4	3	2	1	0
LLT							
R/W-0h							

**Table 17-57. SDCMPL4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	LLT	R/W	0h	Unsigned low-level threshold for the comparator filter output. Reset type: SYSRSn

### 17.8.2.47 SDCPARM4 Register (Offset = 45h) [reset = 0h]

SDCPARM4 is shown in [Figure 17-58](#) and described in [Table 17-58](#).

Return to the [Summary Table](#).

Comparator Filter Parameter Register for Ch4

**Figure 17-58. SDCPARM4 Register**

15	14	13	12	11	10	9	8	
RESERVED		CEN	RESERVED		HZEN	MFIE	CS1_CS0	
		R/W-0h			R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0	
CS1_CS0	IEL	IEH	COSR					
R/W-0h	R/W-0h	R/W-0h	R/W-0h					

**Table 17-58. SDCPARM4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R=0	0h	Reserved
13	CEN	R/W	0h	Comparator Filter enable 0: Disable comparator filter 1: Enable comparator filter Reset type: SYSRSn
12-11	RESERVED	R=0	0h	Reserved
10	HZEN	R/W	0h	High level (Z) Threshold crossing output enable 0: Disable Higher level Threshold (Z) crossing 1: Enable Higher level Threshold (Z) crossing Reset type: SYSRSn
9	MFIE	R/W	0h	Modulator Failure Interrupt Enable 0: Disable modulator failure interrupt and its flag 1: Enable modulator failure interrupt and its flag Reset type: SYSRSn
8-7	CS1_CS0	R/W	0h	Comparator filter structure 00: Comparator filter runs with a sincfast structure 01: Comparator filter runs with a Sinc1 structure 10: Comparator filter runs with a Sinc2 structure 11: Comparator filter runs with a Sinc3 structure Reset type: SYSRSn
6	IEL	R/W	0h	Low-level interrupt enable 0: Disable Lower Threshold interrupt 1: Enable Lower Threshold interrupt Reset type: SYSRSn
5	IEH	R/W	0h	High-level interrupt enable 0: Disable Higher Threshold interrupt 1: Enable Higher Threshold interrupt Reset type: SYSRSn

**Table 17-58. SDCPARM4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	COSR	R/W	0h	<p>Comparator Oversampling ratio.</p> <p>The actual rate is COSR + 1.</p> <p>These bits set the oversampling ratio of the filter.</p> <p>0x1F represents an oversampling ratio of 32</p> <p>Reset type: SYSRSn</p>



**17.8.2.48 SDDATA4 Register (Offset = 46h) [reset = 0h]**

SDDATA4 is shown in [Figure 17-59](#) and described in [Table 17-59](#).

Return to the [Summary Table](#).

Data Filter Data Register (16 or 32bit) for Ch4

**Figure 17-59. SDDATA4 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
DATA32HI																DATA16															
R-0h																R-0h															

**Table 17-59. SDDATA4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode, 16-bit Data in 16b mode Reset type: SYSRSn
15-0	DATA16	R	0h	Lo-order 16b in 32b mode Reset type: SYSRSn

**17.8.2.49 SDDATFIFO4 Register (Offset = 48h) [reset = 0h]**

SDDATFIFO4 is shown in [Figure 17-60](#) and described in [Table 17-60](#).

Return to the [Summary Table](#).

Filter Data FIFO Output(32b) for Ch4

**Figure 17-60. SDDATFIFO4 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
DATA32HI																DATA16															
R-0h																R-0h															

**Table 17-60. SDDATFIFO4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode, 16-bit Data in 16b mode Reset type: SYSRSn
15-0	DATA16	R	0h	Lo-order 16b in 32b mode Reset type: SYSRSn

**17.8.2.50 SDCDATA4 Register (Offset = 4Ah) [reset = 0h]**

SDCDATA4 is shown in [Figure 17-61](#) and described in [Table 17-61](#).

Return to the [Summary Table](#).

Comparator Filter Data Register (16b) for Ch4

**Figure 17-61. SDCDATA4 Register**

15	14	13	12	11	10	9	8
DATA16							
R-0h							
7	6	5	4	3	2	1	0
DATA16							
R-0h							

**Table 17-61. SDCDATA4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DATA16	R	0h	Comparator Data output - 16b only Reset type: SYSRSn

**17.8.2.51 SDCMPHZ4 Register (Offset = 4Ch) [reset = 0h]**

SDCMPHZ4 is shown in [Figure 17-62](#) and described in [Table 17-62](#).

Return to the [Summary Table](#).

High-level (Z) Threshold Register for Ch4

**Figure 17-62. SDCMPHZ4 Register**

15	14	13	12	11	10	9	8
RESERVED	HLTZ						
R/W-0h							
7	6	5	4	3	2	1	0
HLTZ							
R/W-0h							

**Table 17-62. SDCMPHZ4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R=0	0h	Reserved
14-0	HLTZ	R/W	0h	Unsigned High-level threshold (Z) for the comparator filter output Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator SDCMPHx, it does not have the ability to generate an interrupt. Reset type: SYSRSn

**17.8.2.52 SDFIFOCTL4 Register (Offset = 4Dh) [reset = 0h]**

 SDFIFOCTL4 is shown in [Figure 17-63](#) and described in [Table 17-63](#).

 Return to the [Summary Table](#).

FIFO Control Register for Ch4

**Figure 17-63. SDFIFOCTL4 Register**

15		14		13		12		11		10		9		8	
OVFIEN		DRINTSEL		FFEN		FFIEN		RESERVED				SDFFST			
R/W-0h		R/W-0h		R/W-0h		R/W-0h						R-0h			
7		6		5		4		3		2		1		0	
		SDFFST		RESERVED						SDFFIL					
		R-0h								R/W-0h					

**Table 17-63. SDFIFOCTL4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	OVFIEN	R/W	0h	SDFIFO Overflow interrupt enable 0: SDFIFO Overflow condition will not generate an interrupt 1: SDFIFO overflow condition generates an interrupt on SDy_ERR Reset type: SYSRSn
14	DRINTSEL	R/W	0h	Data-Ready Interrupt (DRINT) source select 0 = AF1 (Select non-FIFO data-ready interrupt) 1 = SDFINT1 (Select FIFO data-ready interrupt) Reset type: SYSRSn
13	FFEN	R/W	0h	SDFIFO Enable 0: Disable FIFO operation 1: Enable FIFO operation Note: When FIFO is disabled, FIFO contents are cleared Reset type: SYSRSn
12	FFIEN	R/W	0h	SDFIFO data ready Interrupt Enable Reset type: SYSRSn
11	RESERVED	R=0	0h	Reserved
10-6	SDFFST	R	0h	SDFIFO Status 00000 FIFO empty 00001 FIFO has 1 word ... 10000 FIFO has 16 words Reset type: SYSRSn
5	RESERVED	R=0	0h	Reserved
4-0	SDFFIL	R/W	0h	SDFIFO interrupt level bits The FIFO will generate an interrupt when the FIFO status (SDFFST) >= FIFO level (SDFFIL) Reset type: SYSRSn

**17.8.2.53 SDSYNC4 Register (Offset = 4Eh) [reset = 400h]**

SDSYNC4 is shown in [Figure 17-64](#) and described in [Table 17-64](#).

Return to the [Summary Table](#).

SD Filter Sync control for Ch4

**Figure 17-64. SDSYNC4 Register**

15	14	13	12	11	10	9	8
RESERVED					WTSCLEN	FFSYNCLREN	WTSYNCLR
					R/W-1h	R/W-0h	R=0/W-0h
7	6	5	4	3	2	1	0
WTSYNFLG	WTSYNCEN	SYNCSEL					
R-0h	R/W-0h	R/W-0h					

**Table 17-64. SDSYNC4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R=0	0h	Reserved
10	WTSCLEN	R/W	1h	WTSYNFLG Clear-on-FIFOINT Enable 0: WTSYNFLG can only be cleared manually (using WTSYNCLR bit) 1: WTSYNFLG is cleared automatically on SDFINT Reset type: SYSRSn
9	FFSYNCLREN	R/W	0h	FIFO Clear-on-SDSYNC Enable 0: SDFIFO is not automatically cleared upon receiving SDSYNC 1: SDFIFO is automatically cleared upon receiving SDSYNC Reset type: SYSRSn
8	WTSYNCLR	R=0/W	0h	Wait-for-Sync Flag Clear (always reads 0) 0: Write of 0 has no affect 1: Write of 1 clears WTSYNFLG Reset type: SYSRSn
7	WTSYNFLG	R	0h	Wait-for-Sync Flag 0: SDSYNC event has not occurred 1: SDSYNC event occurred. Reset type: SYSRSn
6	WTSYNCEN	R/W	0h	Wait-for-Sync Enable 0: Incoming Data written to SDFIFO on every Data-Ready (DR) Event 1: Incoming Data written to SDFIFO on DR event only after SDSYNC event occurs Reset type: SYSRSn
5-0	SYNCSEL	R/W	0h	Defines source for the SDSYNC Input on this channel Refer SDSYNcx.SYNCSEL table Reset type: SYSRSn

### 17.8.3 Register to Driverlib Function Mapping

**Table 17-65. SDFM Registers to Driverlib Functions**

File	Driverlib Function
<b>SDIFLG</b>	
sdfm.h	SDFM_getThresholdStatus
sdfm.h	SDFM_getModulatorStatus
sdfm.h	SDFM_getNewFilterDataStatus
sdfm.h	SDFM_getFIFOOverflowStatus
sdfm.h	SDFM_getFIFOISRStatus
sdfm.h	SDFM_getIsrcStatus
sdfm.h	SDFM_clearInterruptFlag
<b>SDIFLGCLR</b>	
sdfm.h	SDFM_clearInterruptFlag
<b>SDCTL</b>	
sdfm.h	SDFM_clearZeroCrossTripStatus
sdfm.h	SDFM_setupModulatorClock
sdfm.h	SDFM_enableMasterInterrupt
sdfm.h	SDFM_disableMasterInterrupt
<b>SDMFILEN</b>	
sdfm.h	SDFM_enableMasterFilter
sdfm.h	SDFM_disableMasterFilter
<b>SDSTATUS</b>	
sdfm.h	SDFM_getZeroCrossTripStatus
<b>SDCTLPARM1</b>	
sdfm.h	SDFM_setupModulatorClock
<b>SDDFPARM1</b>	
sdfm.h	SDFM_enableExternalReset
sdfm.h	SDFM_disableExternalReset
sdfm.h	SDFM_enableFilter
sdfm.h	SDFM_disableFilter
sdfm.h	SDFM_setFilterType
sdfm.h	SDFM_setFilterOverSamplingRatio
sdfm.h	SDFM_enableInterrupt
sdfm.h	SDFM_disableInterrupt
<b>SDDPARM1</b>	
sdfm.h	SDFM_setOutputDataFormat
sdfm.h	SDFM_setDataShiftValue
<b>SDCMPH1</b>	
sdfm.h	SDFM_setCompFilterHighThreshold
<b>SDCMPL1</b>	
sdfm.h	SDFM_setCompFilterLowThreshold
<b>SDCPARM1</b>	
sdfm.h	SDFM_enableComparator
sdfm.h	SDFM_disableComparator
sdfm.h	SDFM_enableZeroCrossEdgeDetect
sdfm.h	SDFM_disableZeroCrossEdgeDetect
sdfm.h	SDFM_enableInterrupt
sdfm.h	SDFM_disableInterrupt
sdfm.h	SDFM_setComparatorFilterType

**Table 17-65. SDFM Registers to Driverlib Functions (continued)**

File	Driverlib Function
sdfm.h	SDFM_setCompFilterOverSamplingRatio
<b>SDDATA1</b>	
sdfm.h	SDFM_getFilterData
<b>SDDATFIFO1</b>	
sdfm.h	SDFM_getFIFOData
<b>SDCDATA1</b>	
sdfm.h	SDFM_getComparatorSincData
<b>SDCMPHZ1</b>	
sdfm.h	SDFM_setCompFilterZeroCrossThreshold
<b>SDFIFOCTL1</b>	
sdfm.h	SDFM_enableFIFOBuffer
sdfm.h	SDFM_disableFIFOBuffer
sdfm.h	SDFM_enableInterrupt
sdfm.h	SDFM_disableInterrupt
sdfm.h	SDFM_getFIFODataCount
sdfm.h	SDFM_setFIFOInterruptLevel
sdfm.h	SDFM_setDataReadyInterruptSource
<b>SDSYNC1</b>	
sdfm.h	SDFM_getWaitForSyncStatus
sdfm.h	SDFM_clearWaitForSyncFlag
sdfm.h	SDFM_enableWaitForSync
sdfm.h	SDFM_disableWaitForSync
sdfm.h	SDFM_setPWMSyncSource
sdfm.h	SDFM_setFIFOClearOnSyncMode
sdfm.h	SDFM_setWaitForSyncClearMode
<b>SDCTLPARM2</b>	
-	See SDCTLPARM1
<b>SDDFPARM2</b>	
-	See SDDFPARM1
<b>SDDPARM2</b>	
-	See SDDPARM1
<b>SDCMPH2</b>	
-	See SDCMPH1
<b>SDCMPL2</b>	
-	See SDCMPL1
<b>SDCPARM2</b>	
-	See SDCPARM1
<b>SDDATA2</b>	
-	See SDDATA1
<b>SDCTLPARM3</b>	
-	See SDCTLPARM1
<b>SDDFPARM3</b>	
-	See SDDFPARM1
<b>SDDPARM3</b>	
-	See SDDPARM1
<b>SDCMPH3</b>	
-	See SDCMPH1



**Table 17-65. SDFM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>SDCMPL3</b>	
-	See SDCMPL1
<b>SDCPARM3</b>	
-	See SDCPARM1
<b>SDDATA3</b>	
-	See SDDATA1
<b>SDCTLPARM4</b>	
-	See SDCTLPARM1
<b>SDDFPARM4</b>	
-	See SDDFPARM1
<b>SDDPARM4</b>	
-	See SDDPARM1
<b>SDCMPH4</b>	
-	See SDCMPH1
<b>SDCMPL4</b>	
-	See SDCMPL1
<b>SDCPARM4</b>	
-	See SDCPARM1
<b>SDDATA4</b>	
-	See SDDATA1

## **Enhanced Pulse Width Modulator (ePWM)**

The enhanced pulse width modulator (ePWM) peripheral is a key element in controlling many of the power electronic systems found in both commercial and industrial equipment. These systems include digital motor control, switch mode power supply control, uninterruptible power supplies (UPS), and other forms of power conversion. The ePWM peripheral performs a digital to analog (DAC) function, where the duty cycle is equivalent to a DAC analog value; it is sometimes referred to as a power DAC.

This chapter is applicable for ePWM type 4 with added register protection capability. See the [TMS320x28xx, 28xxx DSP Peripheral Reference Guide](#) for a list of all devices with an ePWM module of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

Further information about this device can be found in the following document(s):

- [Flexible PWMs Enable Multi-Axis Drives, Multi-Level Inverters](#)

Topic	Page
<b>18.1 Introduction</b> .....	<b>1811</b>
<b>18.2 Configuring Device Pins</b> .....	<b>1818</b>
<b>18.3 ePWM Modules Overview</b> .....	<b>1818</b>
<b>18.4 Time-Base (TB) Submodule</b> .....	<b>1820</b>
<b>18.5 Counter-Compare (CC) Submodule</b> .....	<b>1831</b>
<b>18.6 Action-Qualifier (AQ) Submodule</b> .....	<b>1837</b>
<b>18.7 Dead-Band Generator (DB) Submodule</b> .....	<b>1851</b>
<b>18.8 PWM Chopper (PC) Submodule</b> .....	<b>1857</b>
<b>18.9 Trip-Zone (TZ) Submodule</b> .....	<b>1861</b>
<b>18.10 Event-Trigger (ET) Submodule</b> .....	<b>1866</b>
<b>18.11 Digital Compare (DC) Submodule</b> .....	<b>1872</b>
<b>18.12 ePWM X-BAR</b> .....	<b>1880</b>
<b>18.13 Applications to Power Topologies</b> .....	<b>1884</b>
<b>18.14 Register Lock Protection</b> .....	<b>1901</b>
<b>18.15 High-Resolution Pulse Width Modulator (HRPWM)</b> .....	<b>1903</b>
<b>18.16 ePWM Registers</b> .....	<b>1930</b>

## 18.1 Introduction

This chapter includes an overview and information about each submodule:

- Time-Base Submodule
- Counter Compare Submodule
- Action Qualifier Submodule
- Dead-Band Generator Submodule
- PWM Chopper (PC) Submodule
- Trip Zone Submodule
- Event Trigger Submodule
- Digital Compare Submodule

The ePWM Type 4 is functionally compatible to Type 2 (a Type 3 does not exist). Type 4 has the following enhancements in addition to the Type 2 features:

- **Register Address Map**

Additional registers are required for new features on ePWM Type 4. The ePWM register address space has been remapped for better alignment and easy usage.

- **Delayed Trip Functionality**

Changes have been added to achieve deadband insertion capabilities to support, for example, delayed trip functionality needed for peak current mode control type application scenarios. This has been accomplished by allowing comparator events to go into the Action Qualifier as a trigger event ( Events T1 and T2 ). If comparator T1 / T2 events are used to edit the PWM , changes to the PWM waveform will not take place immediately . Instead , they will synchronize to the next TBCLK.

- **Dead-Band Generator Submodule Enhancements**

Shadowing of the DBCTL register to allow dynamic configuration changes.

- **One Shot and Global Load of Registers**

The ePWM Type 4 allows one shot and global load capability from shadow to active registers to avoid partial loads in, for example, multi-phase applications. It also allows a programmable prescale of shadow to active load events.

- **Trip Zone Submodule Enhancements**

Independent flags have been added to reflect the trip status for each of the TZ sources. Changes have been made to the trip zone submodule to support certain power converter switching techniques like valley switching.

- **Digital Compare Submodule Enhancements**

Blanking window filter register width has been increased from 8 to 16 bits. DCCAP functionality has been enhanced to provide more programmability.

- **PWM SYNC Related Enhancements**

The ePWM Type 4 allows PWM SYNCOUT generation based on CMPC and CMPD events. These events can also be used for PWMSYNC pulse selection.

The ePWM Type 2 is fully compatible to Type 1. Type 2 has the following enhancements in addition to the Type 1 features:

- **High Resolution Dead-Band Capability**

High resolution capability is added to dead-band RED and FED in half-cycle clocking mode.

- **Dead-Band Generator Submodule Enhancements**

The ePWM Type 2 has features to enable both RED and FED on either PWM outputs. Provides increased dead band with 14-bit counters and dead-band / dead-band high-resolution registers are shadowed

- **High Resolution Extension available on ePWMxB outputs**

Provides the ability to enable high-resolution period and duty cycle control on ePWMxB outputs. This is discussed in more detail in [Section 18.15](#).

- **Counter Compare Submodule Enhancements**

The ePWM Type 2 allows Interrupts and SOC events to be generated by additional counter compares CMPC and CMPD.

- **Event Trigger Submodule Enhancements**

Prescaling logic to issue interrupt requests and ADC start of conversion expanded up to every 15 events. It allows software initialization of event counters on SYNC event.

- **Digital Compare Submodule Enhancements**

Digital Compare Trip Select logic [DCTRISEL] has up to 12 external trip sources selected by the Input X-BAR logic in addition to an ability to OR all of them (up to 14 [external and internal sources]) to create the respective DCxEVTs.

- **Simultaneous Writes to TBPRD and CMPx Registers**

This feature allows writes to TBPRD, CMPA:CMPAHR, CMPB:CMPBHR, CMPC and CMPD of any ePWM module to be tied to any other ePWM module, and also allows all ePWM modules to be tied to a particular ePWM module if desired.

- **Shadow to Active Load on SYNC of TBPRD and CMP Registers**

This feature supports simultaneous writes of TBPRD and CMPA/B/C/D registers.

An effective PWM peripheral must be able to generate complex pulse width waveforms with minimal CPU overhead or intervention. It must be highly programmable and very flexible while being easy to understand and use. The ePWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross coupling or sharing of resources has been avoided; instead, the ePWM is built up from smaller single channel submodules with separate resources that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand its operation quickly.

In this document, the letter x within a signal or submodule name is used to indicate a generic ePWM instance on a device. For example, output signals EPWMxA and EPWMxB refer to the output signals from the ePWMx instance. Thus, EPWM1A and EPWM1B belong to ePWM1 and likewise EPWM4A and EPWM4B belong to ePWM4.

### Type0 to Type1 Enhancements

- **Increased Dead-Band Resolution**

Dead-band clocking has been enhanced to allow half-cycle clocking to double resolution.

- **Enhanced Interrupt and SOC Generation**

Interrupts and ADC start-of-conversion can now be generated on both the TBCTR == zero and TBCTR == period events. This feature enables dual edge PWM control. Additionally, the ADC start-of-conversion can be generated from an event defined in the digital compare submodule.

- **High Resolution Period Capability**

Provides the ability to enable high-resolution period. This is discussed in more detail in the device-specific *High-Resolution Pulse Width Modulator (HRPWM)* document..

- **Digital Compare Submodule**

The digital compare submodule enhances the event triggering and trip zone submodules by providing filtering, blanking and improved trip functionality to digital compare signals. Such features are essential for peak current mode control and for support of analog comparators.

---

**NOTE:** The name of the sync signal that goes to the CMPSS and GPDAC has been updated from PWMSYNC to EPWMSYNCPER (SYNCPER/PWMSYNCPER/EPWMxSYNCPER) to avoid confusion with the other EPWM sync signals EPWMSYNCI and EPWMSYNCO. For a description of what these signals are, see [Table 18-2](#).

---

### 18.1.1 Submodule Overview

The ePWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. Multiple ePWM modules are instanced within a device as shown in [Figure 18-1](#). Each ePWM instance is identical with one exception. Some instances include a hardware extension that allows more precise control of the PWM outputs. This extension is the high-resolution pulse width modulator (HRPWM) and is described in [Section 18.15](#). See the device-specific data manual to determine which ePWM instances include this feature. Each ePWM module is indicated by a numerical value starting with 1. For example ePWM1 is the first instance and ePWM3 is the third instance in the system and ePWMx indicates any instance.

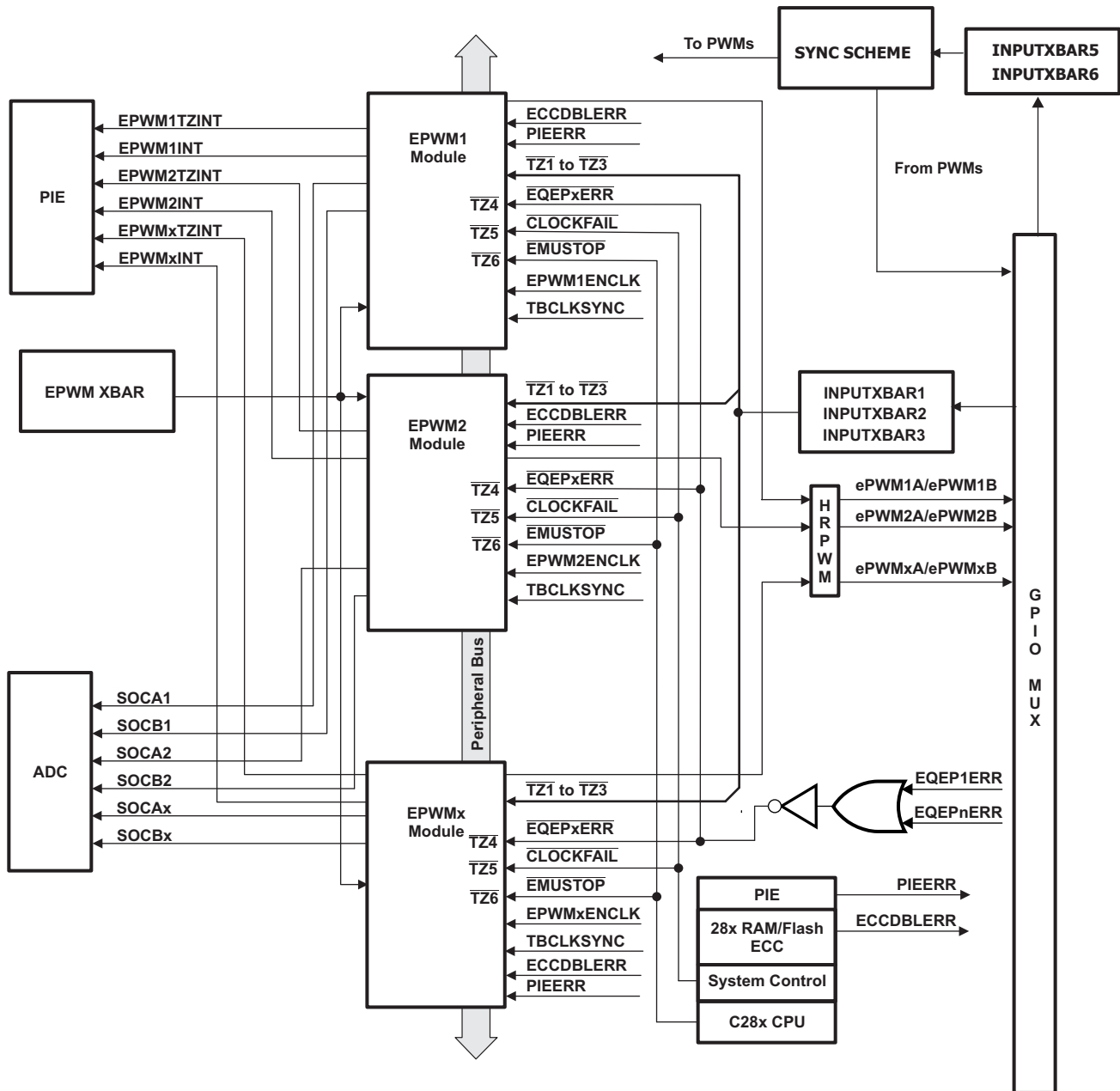
The ePWM modules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the capture peripheral submodules (eCAP). The number of submodules is device-dependent and based on target application needs. Submodules can also operate standalone.

Each ePWM module supports the following features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software.
- Programmable phase-control support for lag or lead operation relative to other ePWM modules.
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis.
- Dead-band generation with independent rising and falling edge delay control.
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions.
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs.
- All events can trigger both CPU interrupts and ADC start of conversion (SOC)
- Programmable event prescaling minimizes CPU overhead on interrupts.
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives.

Each ePWM module is connected to the input/output signals shown in [Figure 18-1](#). The signals are described in detail in subsequent sections.

Figure 18-1. Multiple ePWM Modules



Copyright © 2017, Texas Instruments Incorporated

A This signal exists only on devices with an eQEP submodule.

The order in which the ePWM modules are connected may differ from what is shown in Figure 18-1. See Section 18.4.3.3 for the synchronization scheme for a particular device. Each ePWM module consists of eight submodules and is connected within a system via the signals shown in Figure 18-2.

Figure 18-2. Submodules and Signal Connections for an ePWM Module

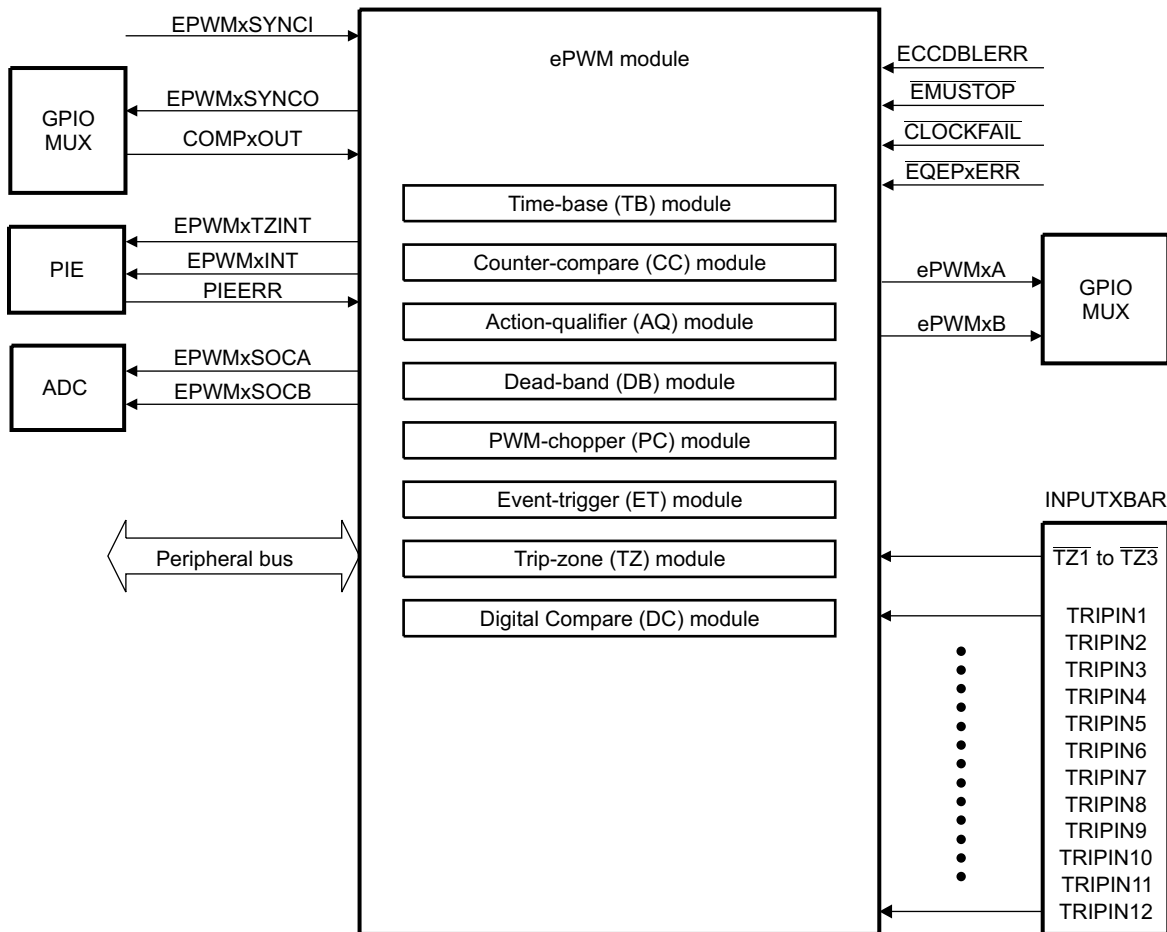


Figure 18-3 shows more internal details of a single ePWM module. The main signals used by the ePWM module are:

- **PWM output signals (EPWMxA and EPWMxB).**

The PWM output signals are made available external to the device through the GPIO peripheral described in the *System Control and Interrupts* chapter for your device.

- **Trip-zone signals (TZ1 to TZ6).**

These input signals alert the ePWM module of fault conditions external to the ePWM module. Each submodule on a device can be configured to either use or ignore any of the trip-zone signals. The TZ1 to TZ3 trip-zone signals can be configured as asynchronous inputs through the GPIO peripheral using the Input X-BAR logic, refer to Figure 18-49. TZ4 is connected to an inverted EQEPx error signal (EQEPxERR), which can be generated from any one of the EQEP submodule (for those devices with an EQEP module). TZ5 is connected to the system clock fail logic, and TZ6 is connected to the EMUSTOP output from the CPU. This allows you to configure a trip action when the clock fails or the CPU halts.

- **Time-base synchronization input (EPWMxSYNCl), output (EPWMxSYNCO) and peripheral (EPWMxSYNCPER) signals.**

The synchronization signals daisy chain the ePWM module together. Each module can be configured via INPUTXBAR6 to either use or ignore its synchronization input. The clock synchronization input and output signal are brought out to pins only for ePWM1 (ePWM module #1). The ePWM module are separate into groups of three for syncing purposes. An external sync signal (EXTSYNClN1 or EXTSYNClN2) may be used to issue a sync signal to the first ePWM module in each chain. These same module can also send their EPWMxSYNCO signal to a GPIO. For more information, see Section 18.4.3.3.

Each ePWM module also generates another PWMSYNC signal called EPWMxSYNCPER. EPWMxSYNCPER goes to the GPDAC and CMPSS for synchronization purposes. It is configured using the HRPCTL register but has no relation with the HRPWM. For more information on how EPWMxSYNCPER is used by the GPDAC and CMPSS, see their respective chapters.

- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB).**

Each ePWM module has two ADC start of conversion signals. Any ePWM module can trigger a start of conversion. Whichever event triggers the start of conversion is configured in the event-trigger submodule of the ePWM.

- **Comparator output signals (COMPxOUT).**

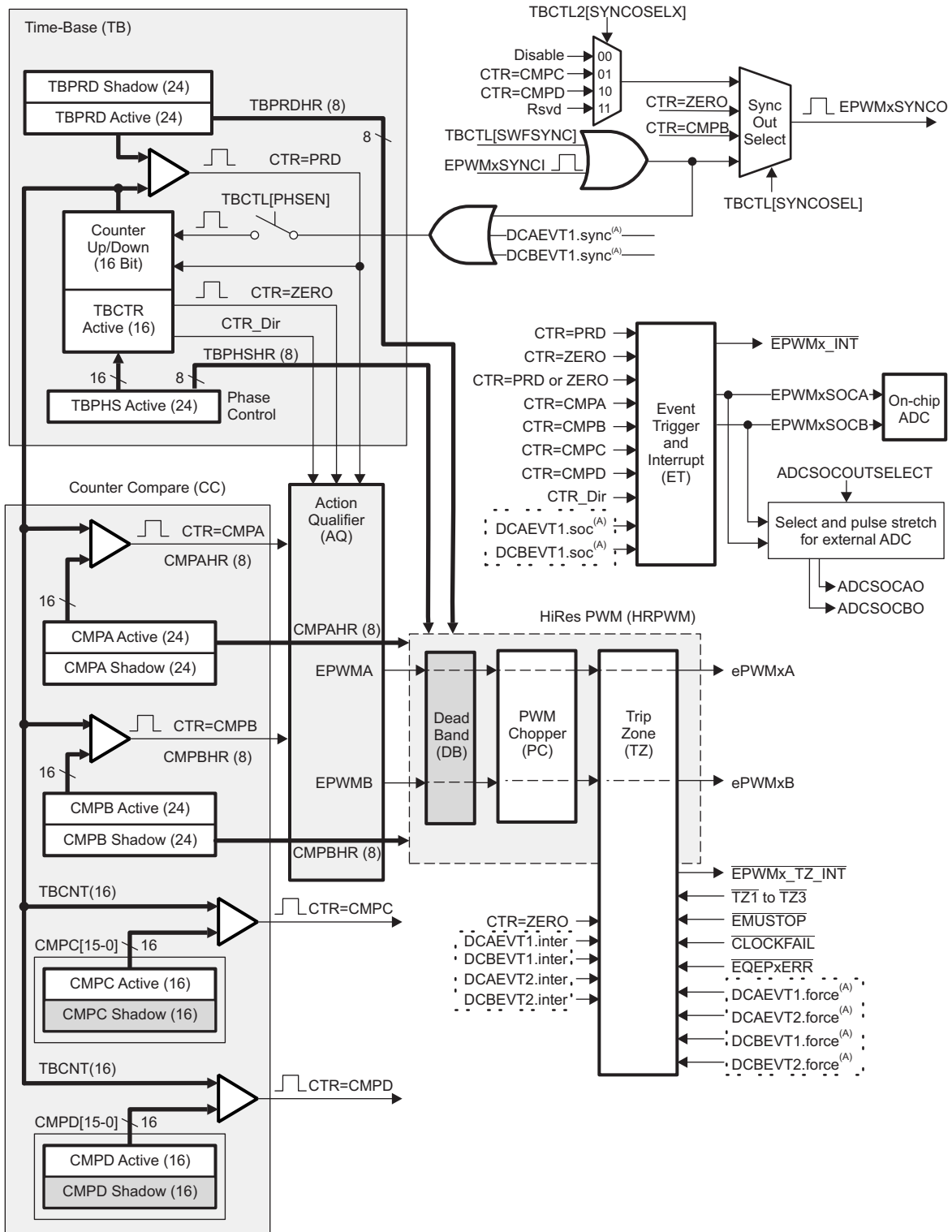
Output signals from the comparator module can be fed through the Input X-BAR to one or all of the 12 trip inputs [TRIPIN1 - TRIPIN12] and in conjunction with the trip zone signals can generate digital compare events.

- **Peripheral Bus**

The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the ePWM register file.



Figure 18-3. ePWM modules and Critical Internal Signal Interconnects



Copyright © 2017, Texas Instruments Incorporated

A --\*These events are generated by the ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs.

## 18.2 Configuring Device Pins

To connect the device input pins to the module, the Input X-BAR must be used. Some examples of when an external signal may be needed are TZx, TRIPx, and EXTSYNCCIN. Any GPIO on the device can be configured as an input. The GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSEL register bits to 11b. The internal pullups can be configured in the GPyPUD register. Since the GPIO mode is used, the GPyINV register can invert the signals. Additionally, some TRIPx (TRIP4-12 excluding TRIP6) signals must be routed through the ePWM X-Bar in addition to the Input X-Bar.

The GPIO mux registers must be configured for this peripheral. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

See the *GPIO* chapter for more details on GPIO mux, GPIO settings, and XBAR configuration.

## 18.3 ePWM Modules Overview

Eight submodules are included in every ePWM peripheral. Each of these submodules performs specific tasks that can be configured by software.

[Table 18-1](#) lists the eight key submodules together with a list of their main configuration parameters. For example, if you need to adjust or control the duty cycle of a PWM waveform, then you should see the counter-compare submodule in [Section 18.5](#) for relevant details.

**Table 18-1. Submodule Configuration Parameters**

Submodule	Configuration Parameter or Option
Time Base (TB)	<ul style="list-style-type: none"> <li>• Scale the time-base clock (TBCLK) relative to the ePWM clock (EPWMCLK).</li> <li>• Configure the PWM time-base counter (TBCTR) frequency or period.</li> <li>• Set the mode for the time-base counter:                             <ul style="list-style-type: none"> <li>– count-up mode: used for asymmetric PWM</li> <li>– count-down mode: used for asymmetric PWM</li> <li>– count-up-and-down mode: used for symmetric PWM</li> </ul> </li> <li>• Configure the time-base phase relative to another ePWM module.</li> <li>• Synchronize the time-base counter between modules through hardware or software.</li> <li>• Configure the direction (up or down) of the time-base counter after a synchronization event.</li> <li>• Simultaneous writes to the TBPRD registers on all PWM's corresponding to the configuration on EPWMLINK.</li> <li>• Configure how the time-base counter will behave when the device is halted by an emulator.</li> <li>• Specify the source for the synchronization output of the ePWM module                             <ul style="list-style-type: none"> <li>– Synchronization input signal</li> <li>– Time-base counter equal to zero</li> <li>– Time-base counter equal to counter-compare B (CMPB)</li> <li>– No output synchronization signal generated.</li> </ul> </li> <li>• Configure one shot and global load of registers in this module.</li> </ul>
Counter Compare (CC)	<ul style="list-style-type: none"> <li>• Specify the PWM duty cycle for output EPWMxA and/or output EPWMxB</li> <li>• Specify the time at which switching events occur on the EPWMxA or EPWMxB output</li> <li>• Specify the programmable delay for interrupt and SOC generation with additional comparators</li> <li>• Simultaneous writes to the CMPA, CMPB, CMPC, CMPD registers on all PWM's corresponding to the configuration on EPWMLINK.</li> <li>• Configure one shot and global load of registers in this module.</li> </ul>

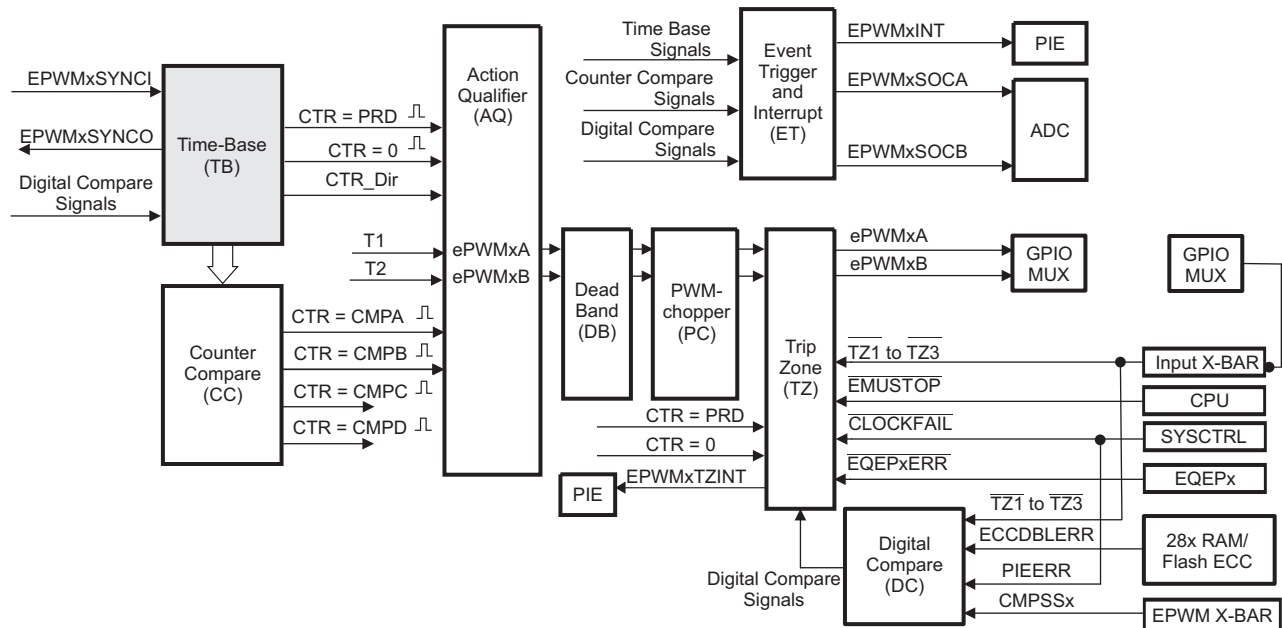
**Table 18-1. Submodule Configuration Parameters (continued)**

Submodule	Configuration Parameter or Option
Action Qualifier (AQ)	<ul style="list-style-type: none"> <li>• Specify the type of action taken when a time-base counter-compare, trip-zone submodule, or comparator event occurs: <ul style="list-style-type: none"> <li>– No action taken</li> <li>– Output EPWMxA and/or EPWMxB switched high</li> <li>– Output EPWMxA and/or EPWMxB switched low</li> <li>– Output EPWMxA and/or EPWMxB toggled</li> </ul> </li> <li>• Force the PWM output state through software control</li> <li>• Configure and control the PWM dead band through software</li> <li>• Configure one shot and global load of registers in this module.</li> </ul>
Dead Band (DB)	<ul style="list-style-type: none"> <li>• Control of traditional complementary dead-band relationship between upper and lower switches</li> <li>• Specify the output rising-edge-delay value</li> <li>• Specify the output falling-edge delay value</li> <li>• Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification.</li> <li>• Option to enable half-cycle clocking for double resolution.</li> <li>• Allow ePWMxB phase shifting with respect to the ePWMxA output.</li> <li>• Configure one shot and global load of registers in this module.</li> </ul>
PWM Chopper (PC)	<ul style="list-style-type: none"> <li>• Create a chopping (carrier) frequency.</li> <li>• Pulse width of the first pulse in the chopped pulse train.</li> <li>• Duty cycle of the second and subsequent pulses.</li> <li>• Bypass the PWM chopper module entirely. In this case the PWM waveform is passed through without modification.</li> </ul>
Trip Zone (TZ)	<ul style="list-style-type: none"> <li>• Configure the ePWM module to react to one, all, or none of the trip-zone signals or digital compare events.</li> <li>• Specify the trip action taken when a fault occurs: <ul style="list-style-type: none"> <li>– Force EPWMxA and/or EPWMxB high</li> <li>– Force EPWMxA and/or EPWMxB low</li> <li>– Force EPWMxA and/or EPWMxB to a high-impedance state</li> <li>– Configure EPWMxA and/or EPWMxB to ignore any trip condition.</li> </ul> </li> <li>• Configure how often the ePWM will react to each trip-zone signal: <ul style="list-style-type: none"> <li>– One-shot</li> <li>– Cycle-by-cycle</li> </ul> </li> <li>• Enable the trip-zone to initiate an interrupt.</li> <li>• Bypass the trip-zone module entirely.</li> <li>• Programmable option for cycle-by-cycle trip clear</li> <li>• If desired, independently configure trip actions taken when time-base counter is counting down.</li> </ul>
Event Trigger (ET)	<ul style="list-style-type: none"> <li>• Enable the ePWM events that will trigger an interrupt.</li> <li>• Enable ePWM events that will trigger an ADC start-of-conversion event.</li> <li>• Specify the rate at which events cause triggers (every occurrence or every 2nd or up to 15th occurrence)</li> <li>• Poll, set, or clear event flags</li> </ul>
Digital Compare (DC)	<ul style="list-style-type: none"> <li>• Enables comparator (COMP) module outputs and trip zone signals which are configured using the Input X-BAR to create events and filtered events</li> <li>• Specify event-filtering options to capture TBCTR counter, generate blanking window, or insert delay in PWM output or time-base counter based on captured value.</li> </ul>

## 18.4 Time-Base (TB) Submodule

Each ePWM module has its own time-base submodule that determines all of the event timing for the ePWM module. Built-in synchronization logic allows the time-base of multiple ePWM modules to work together as a single system. Figure 18-4 illustrates the time-base module's place within the ePWM.

**Figure 18-4. Time-Base Submodule**



### 18.4.1 Purpose of the Time-Base Submodule

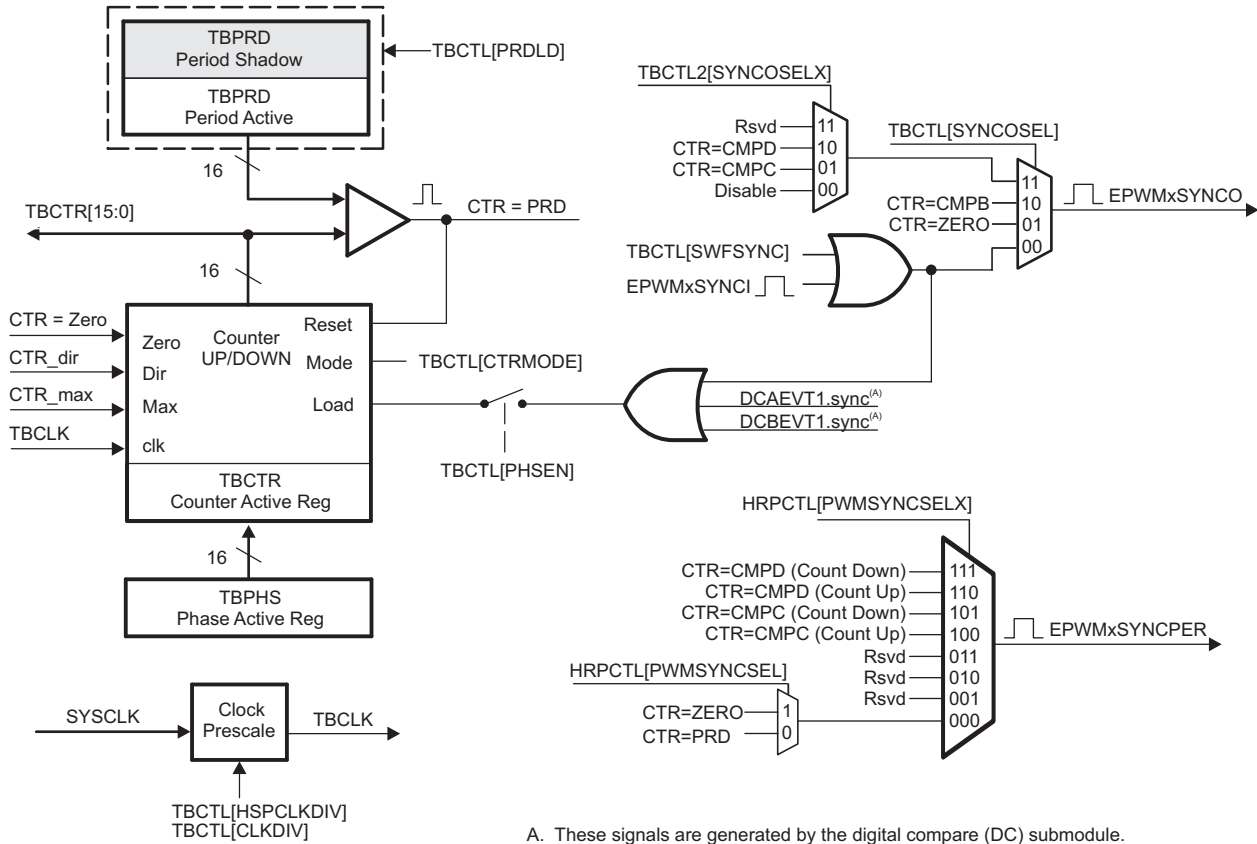
You can configure the time-base submodule for the following:

- Specify the ePWM time-base counter (TBCTR) frequency or period to control how often events occur.
- Manage time-base synchronization with other ePWM modules.
- Maintain a phase relationship with other ePWM modules.
- Set the time-base counter to count-up, count-down, or count-up-and-down mode.
- Generate the following events:
  - CTR = PRD: Time-base counter equal to the specified period (TBCTR = TBPRD).
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00).
- Configure the rate of the time-base clock; a prescaled version of the ePWM clock (EPWMCLK). This allows the time-base counter to increment/decrement at a slower rate.

### 18.4.2 Controlling and Monitoring the Time-Base Submodule

The block diagram in Figure 18-5 shows the critical signals and registers of the time-base submodule. Table 18-2 provides descriptions of the key signals associated with the time-base submodule.

Figure 18-5. Time-Base Submodule Signals and Registers



A. These signals are generated by the digital compare (DC) submodule.

Table 18-2. Key Time-Base Signals

Signal	Description
EPWMxSYNCI	Time-base synchronization input. Input pulse used to synchronize the time-base counter with the counter of ePWM module earlier in the synchronization chain. An ePWM peripheral can be configured to use or ignore this signal. For the first ePWM module in each synchronization chain, this signal may come from a device pin via INPUT5 or INPUT6 of the Input X-BAR or from a previous ePWM module. For subsequent ePWM modules in each chain, this signal is passed from another ePWM peripheral. For example, EPWM2SYNCI is generated by the ePWM1 peripheral, EPWM3SYNCI is generated by ePWM2 and so forth. For information on the synchronization order of a particular device, see Section 18.4.3.3.
EPWMxSYNCO	Time-base synchronization output. This output pulse is used to synchronize the counter of an ePWM module later in the synchronization chain. The ePWM module generates this signal from one of three event sources: 1. EPWMxSYNCI (Synchronization input pulse) 2. CTR = Zero: The time-base counter equal to zero (TBCTR = 0x00). 3. CTR = CMPB: The time-base counter equal to the counter-compare B (TBCTR = CMPB) register.
EPWMxSYNCPER	Time-base peripheral synchronization output. This output signal is used to synchronize the GPDAC and CMPSS to the EPWM. It can be configured using the HRPCTL register. Note that this signal has no relation with the HRPWM.
CTR = PRD	Time-base counter equal to the specified period. This signal is generated whenever the counter value is equal to the active period register value. That is when TBCTR = TBPRD.

**Table 18-2. Key Time-Base Signals (continued)**

Signal	Description
CTR = Zero	Time-base counter equal to zero This signal is generated whenever the counter value is zero. That is when TBCTR equals 0x00.
CTR = CMPB	Time-base counter equal to active counter-compare B register (TBCTR = CMPB). This event is generated by the counter-compare submodule and used by the synchronization out logic
CTR_dir	Time-base counter direction. Indicates the current direction of the ePWM's time-base counter. This signal is high when the counter is increasing and low when it is decreasing.
CTR_max	Time-base counter equal max value. (TBCTR = 0xFFFF) Generated event when the TBCTR value reaches its maximum value. This signal is only used only as a status bit
TBCLK	Time-base clock. This is a prescaled version of the ePWM clock (EPWMCLK) and is used by all submodules within the ePWM. This clock determines the rate at which time-base counter increments or decrements.

### 18.4.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter. Figure 18-6 shows the period ( $T_{pwm}$ ) and frequency ( $F_{pwm}$ ) relationships for the up-count, down-count, and up-down-count time-base counter modes when the period is set to 4 (TBPRD = 4). The time increment for each step is defined by the time-base clock (TBCLK) which is a prescaled version of the ePWM clock (EPWMCLK).

The time-base counter has three modes of operation selected by the time-base control register (TBCTL):

- **Up-Down-Count Mode:**

In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.

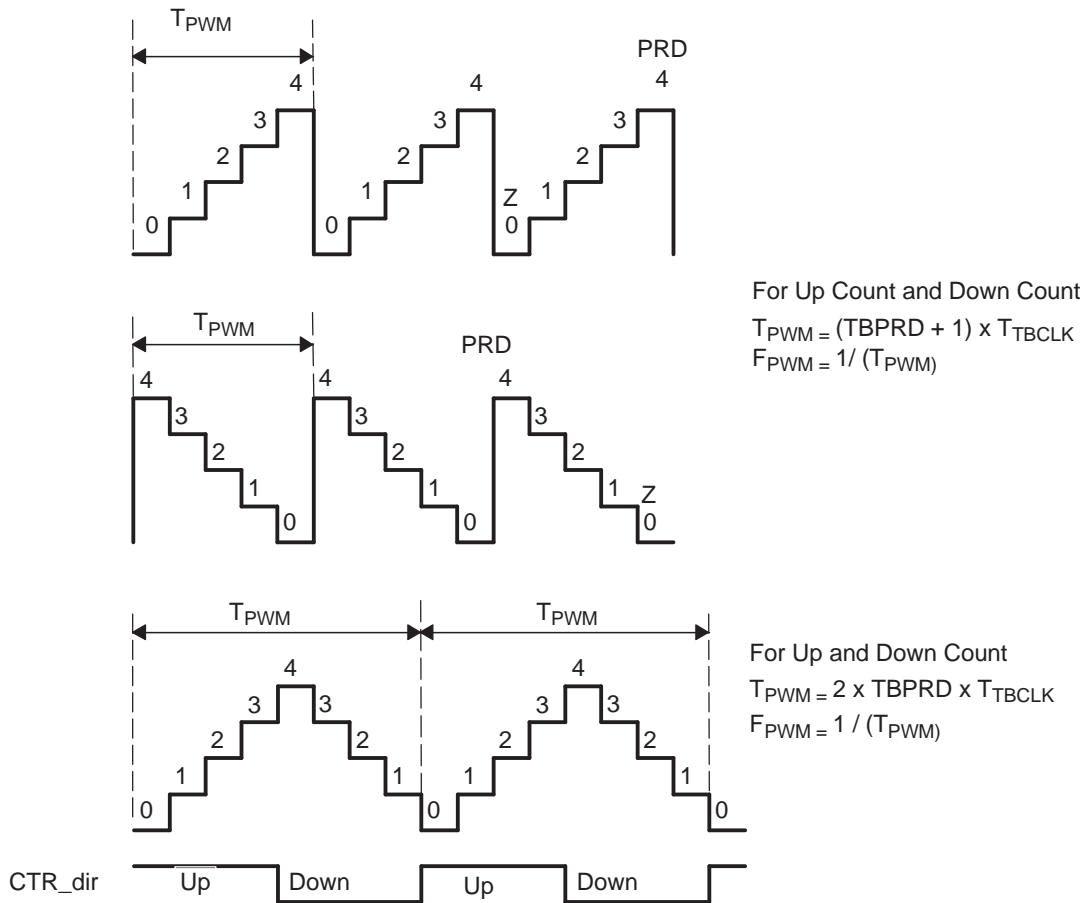
- **Up-Count Mode:**

In this mode, the time-base counter starts from zero and increments until it reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.

- **Down-Count Mode:**

In down-count mode, the time-base counter starts from the period (TBPRD) value and decrements until it reaches zero. When it reaches zero, the time-base counter is reset to the period value and it begins to decrement once again.

Figure 18-6. Time-Base Frequency and Period



### 18.4.3.1 Time-Base Period Shadow Register

The time-base period register (TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the ePWM module:

- **Active Register**

The active register controls the hardware and is responsible for actions that the hardware causes or invokes.

- **Shadow Register**

The shadow register buffers or provides a temporary holding location for the active register. It has no direct effect on any control hardware. At a strategic point in time the shadow register's content is transferred to the active register. This prevents corruption or spurious operation due to the register being asynchronously modified by software.

The memory address of the shadow period register is the same as the active register. Which register is written to or read from is determined by the TBCTL[PRDL] bit. This bit enables and disables the TBPRD shadow register as follows:

- **Time-Base Period Shadow Mode:**

The TBPRD shadow register is enabled when TBCTL[PRDL] = 0. Reads from and writes to the TBPRD memory address go to the shadow register. The shadow register contents are transferred to the active register (TBPRD (Active) ← TBPRD (shadow)) when the time-base counter equals zero (TBCTR = 0x00) and/or a sync event as determined by the TBCTL2[PRDLDSYNC] bit. The PRDLDSYNC bit is valid only if TBCTL[PRDL] = 0. By default the TBPRD shadow register is enabled. The sources for the SYNC input is explained in [Section 18.4.3.3](#).

The global load control mechanism can also be used with the time-base period register by configuring the appropriate bits in the global load configuration register (GLDCFG). When global load mode is selected the transfer of contents from shadow register to active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in Global Shadow to Active Load Control Register (GLDCTL). Global load control mechanism is explained in [Section 18.4.7](#).

- **Time-Base Period Immediate Load Mode:**

If immediate load mode is selected (TBCTL[PRDL] = 1), then a read from or a write to the TBPRD memory address goes directly to the active register.

#### 18.4.3.2 Time-Base Clock Synchronization

The TBCLKSYNC bit in the peripheral clock enable registers allows all users to globally synchronize all enabled ePWM modules to the time-base clock (TBCLK). When set, all enabled ePWM module clocks are started with the first rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescalers for each ePWM module must be set identically.

The proper procedure for enabling ePWM clocks is as follows:

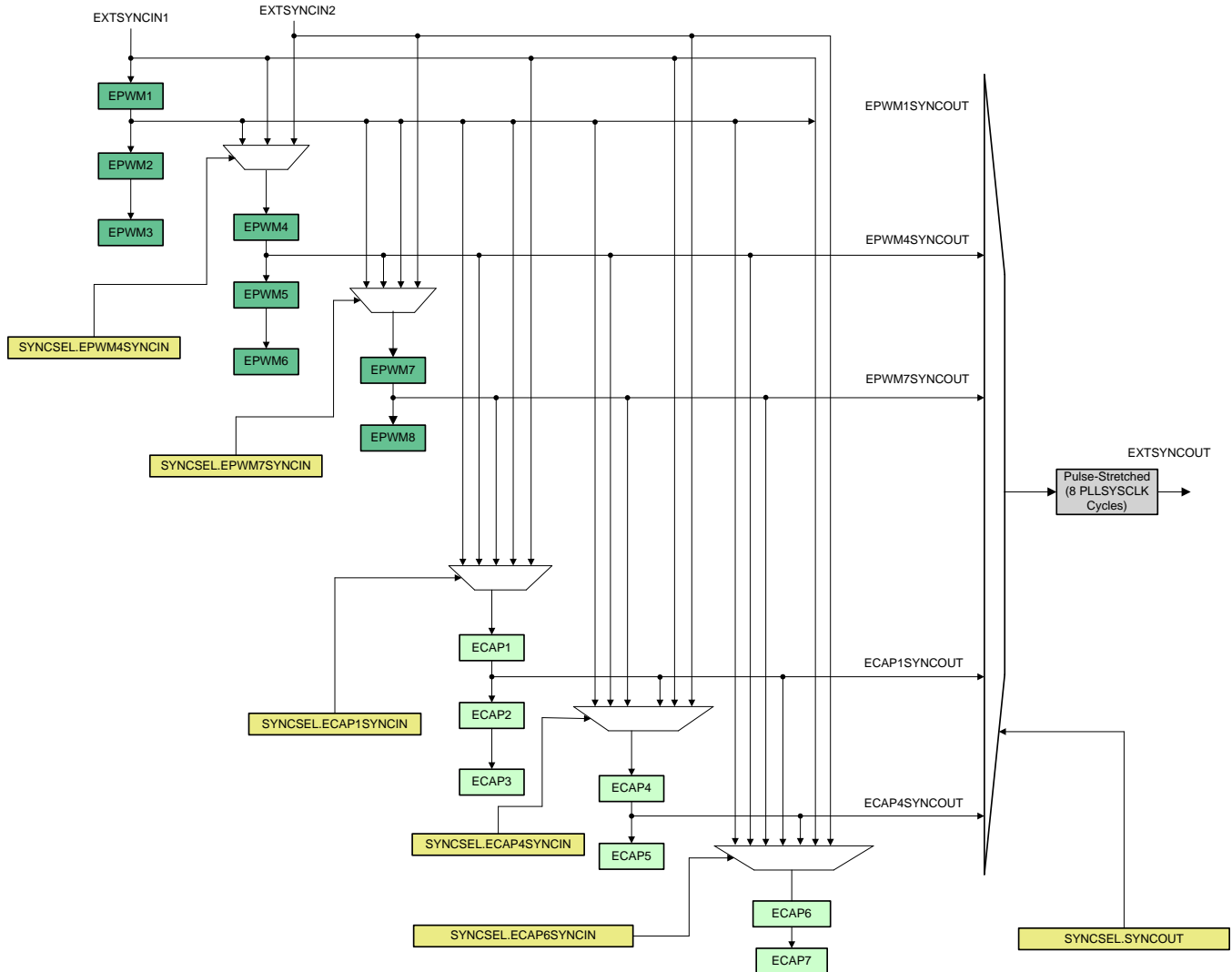
1. Enable ePWM module clocks in the PCLKCRx register
2. Set TBCLKSYNC= 0
3. Configure ePWM modules
4. Set TBCLKSYNC= 1

#### 18.4.3.3 Time-Base Counter Synchronization

The ePWM type 4 introduces a new synchronization scheme that allows for increased flexibility of synchronization of the ePWM modules. Each ePWM module has a synchronization input (SYNCI), a synchronization output (SYNCO) and a peripheral synchronization output (SYNCPER). In [Figure 19-8](#), EXTSYNC1 is sourced from INPUTXBAR5 and EXTSYNC2 is sourced from INPUTXBAR6, which can be configured to select any GPIO as the synchronization input. When configuring the sync chain propagation path using the SYNCSEL registers, make sure that the longest path does not exceed four ePWM/eCAP modules.



Figure 18-7. Time-Base Counter Synchronization Scheme



**NOTE:** See the data manual for the number of ePWM and eCAP modules available on your specific device.

Each ePWM module can be configured to use or ignore the synchronization input. If the TBCTL[PHSEN] bit is set, then the time-base counter (TBCTR) of the ePWM module will be automatically loaded with the phase register (TBPHS) contents when one of the following conditions occur:

- **EPWMxSYNCI: Synchronization Input Pulse:**

The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (TBPHS → TBCTR). This operation occurs on the next valid time-base clock (TBCLK) edge.

The delay from internal master module to slave modules is given by:

- if ( TBCLK = EPWMCLK): 2 x EPWMCLK
- if ( TBCLK < EPWMCLK): 1 x TBCLK

- **Software Forced Synchronization Pulse:**

Writing a 1 to the TBCTL[SWFSYNC] control bit invokes a software forced synchronization. This pulse is ORed with the synchronization input signal, and therefore has the same effect as a pulse on EPWMxSYNCI.

- **Digital Compare Event Synchronization Pulse:**

DCAEVT1 and DCBEVT1 digital compare events can be configured to generate synchronization pulses which have the same affect as EPWMxSYNCl.

This feature enables the ePWM module to be automatically synchronized to the time base of another ePWM module. Lead or lag phase control can be added to the waveforms generated by different ePWM modules to synchronize them. In up-down-count mode, the TBCTL[PSHDIR] bit configures the direction of the time-base counter immediately after a synchronization event. The new direction is independent of the direction prior to the synchronization event. The PSHDIR bit is ignored in count-up or count-down modes. See [Figure 18-8](#) through [Figure 18-11](#) for examples.

Clearing the TBCTL[PHSEN] bit configures the ePWM to ignore the synchronization input pulse. The synchronization pulse can still be allowed to flow-through to the EPWMxSYNCO and be used to synchronize other ePWM modules. In this way, you can set up a master time-base (for example, ePWM1) and downstream modules (ePWM2 - ePWMx) may elect to run in synchronization with the master. See [Section 18.13](#) for more details on synchronization strategies.

#### 18.4.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules

The TBCLKSYNC bit can be used to globally synchronize the time-base clocks of all enabled ePWM modules on a device. This bit is part of the device's clock enable registers and is described in the *System Control and Interrupts* section of this manual. When TBCLKSYNC = 0, the time-base clock of all ePWM modules is stopped (default). When TBCLKSYNC = 1, all ePWM time-base clocks are started with the rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each ePWM module must be set identically. The proper procedure for enabling the ePWM clocks is as follows:

1. Enable the individual ePWM module clocks. This is described in the *System Control and Interrupts* chapter.
2. Set TBCLKSYNC = 0. This will stop the time-base clock within any enabled ePWM module.
3. Configure the prescaler values and desired ePWM modes.
4. Set TBCLKSYNC = 1.

#### 18.4.5 Simultaneous Writes to TBPRD and CMPx Registers Between ePWM Modules

For variable frequency applications, there is a need for simultaneous writes of TBPRD and CMPx registers between ePWM modules. This prevents situations where a CTR = 0 or CTR = PRD pulse forces a shadow to active load of these registers before all registers are updated between ePWM modules (resulting in some registers being loaded from new shadow values while others are loaded from old shadow values). To support this, an ePWM register linking scheme for TBPRD:TBPRDHR, CMPA:CMPAHR, CMPB:CMPBHR, CMPC, and CMPD registers between PWM modules has been added.

For a particular ePWM module # A , user code writes "B+1", to the linked register bit-field in EPWMXLINK. "B" is the ePWM module # being linked to (that is, writes to the ePWM module "B" TBPRD:TBPRDHR, CMPA:CMPAHR, CMPB:CMPBHR, or CMPC will simultaneously be written to corresponding register in ePWM module "A"). For instance if ePWM3 EPWMXLINK register is configured so that CMPA:CMPAHR are linked to ePWM1, then a write to CMPA:CMPAHR in ePWM 1 will simultaneously write the same value to CMPA:CMPAHR in ePWM3. If ePWM4 also has its CMPA:CMPAHR registers linked to ePWM1, then a write to ePWM 1 will write the same value to the CMPA:CMPAHR registers in both ePWM3 and ePWM4.

The register description for EPWMXLINK clearly explains the linked register bit-field values for corresponding ePWM.

#### 18.4.6 Time-Base Counter Modes and Timing Waveforms

The time-base counter operates in one of four modes:

- Up-count mode which is asymmetrical
- Down-count mode which is asymmetrical
- Up-down-count which is symmetrical

- Frozen where the time-base counter is held constant at the current value

To illustrate the operation of the first three modes, the following timing diagrams show when events are generated and how the time-base responds to an EPWMxSYNCl signal.

**Figure 18-8. Time-Base Up-Count Mode Waveforms**

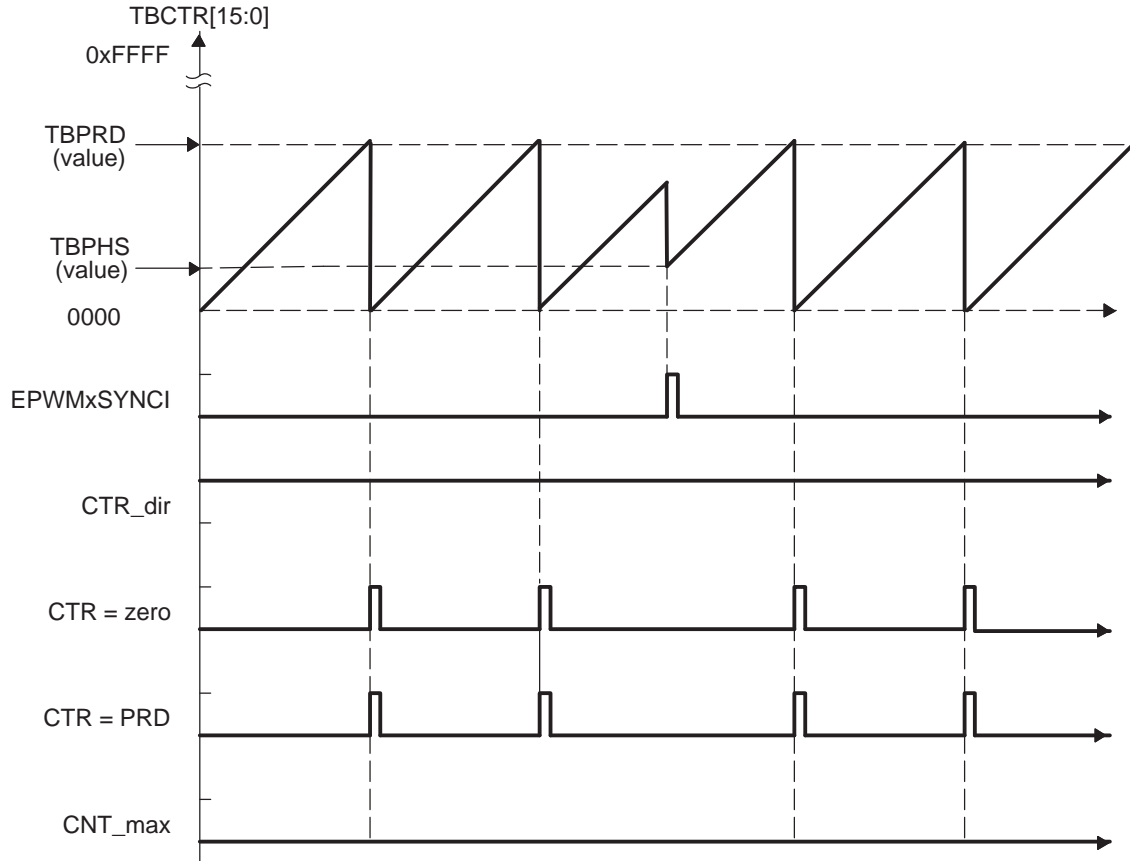


Figure 18-9. Time-Base Down-Count Mode Waveforms

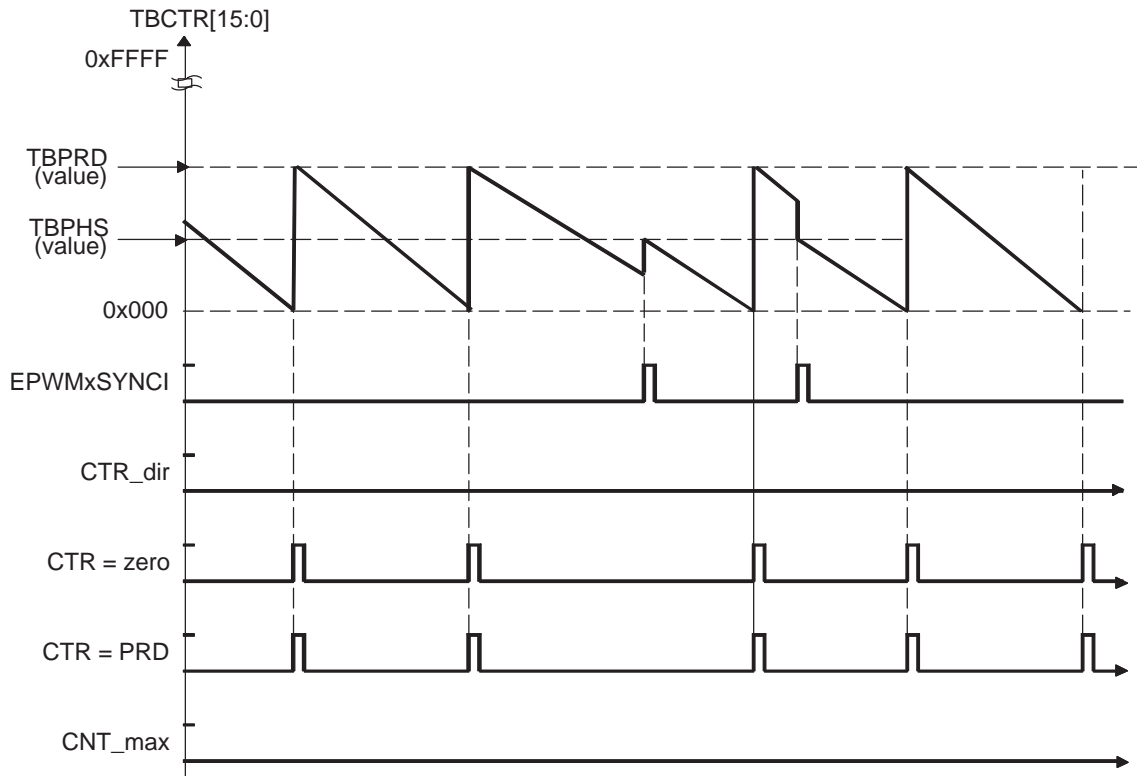
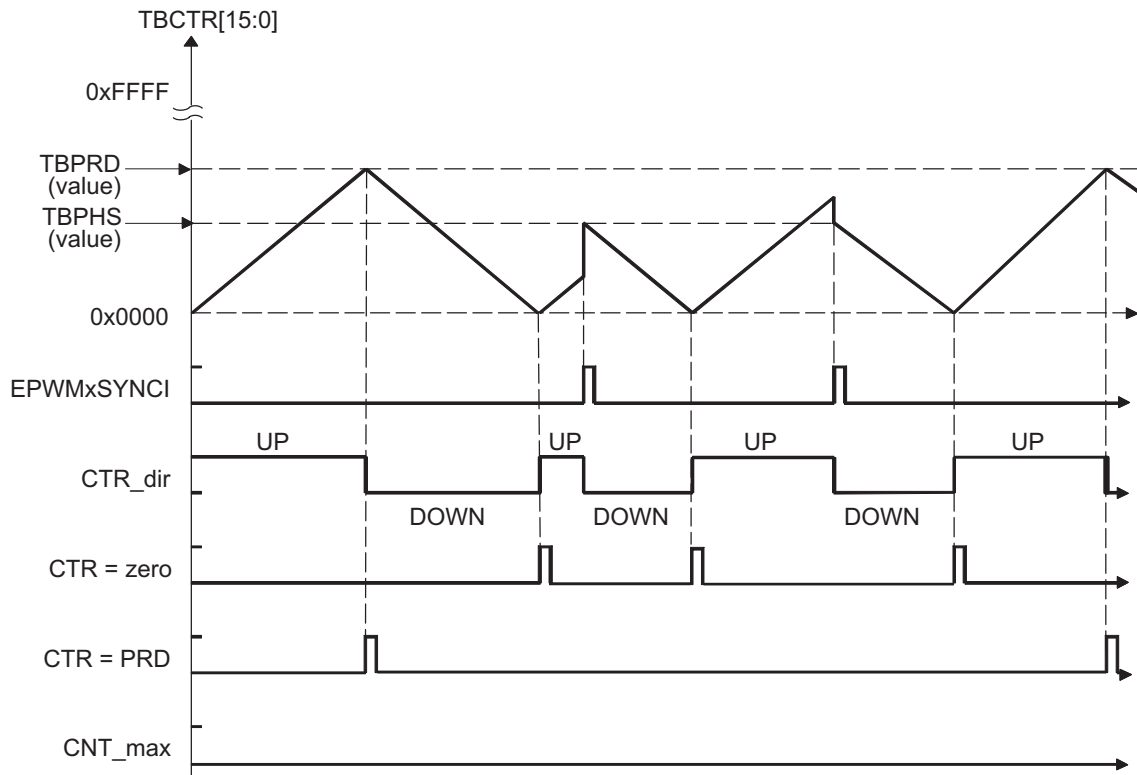
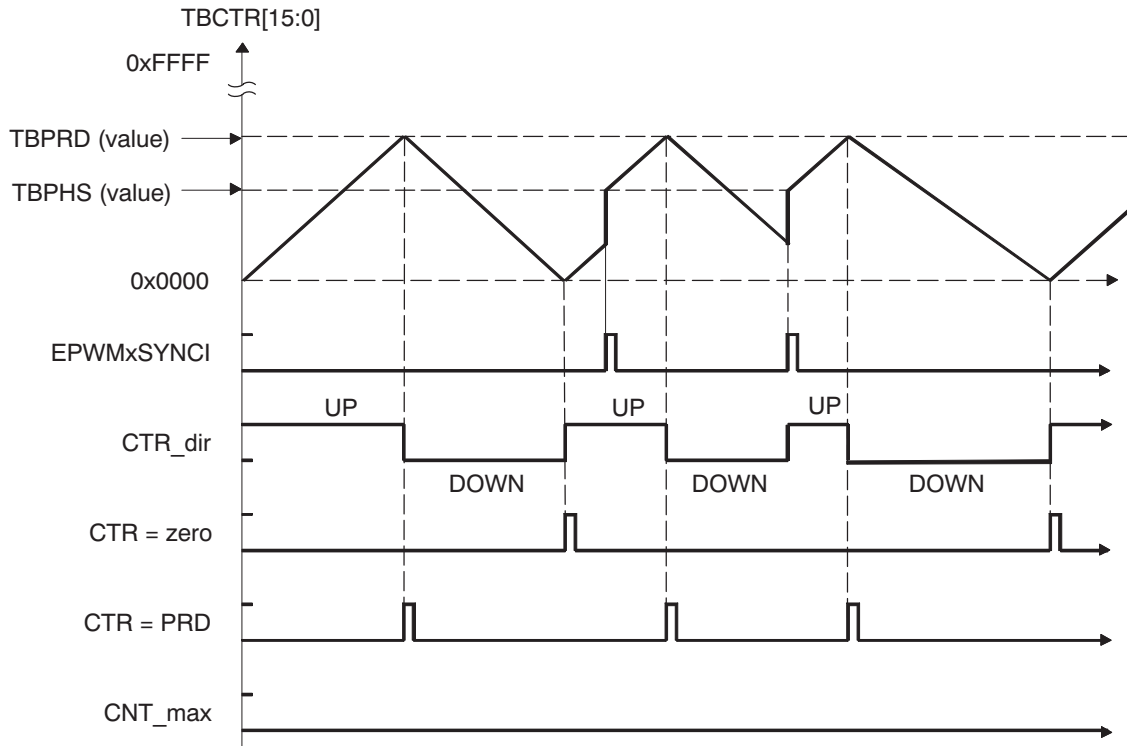


Figure 18-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event

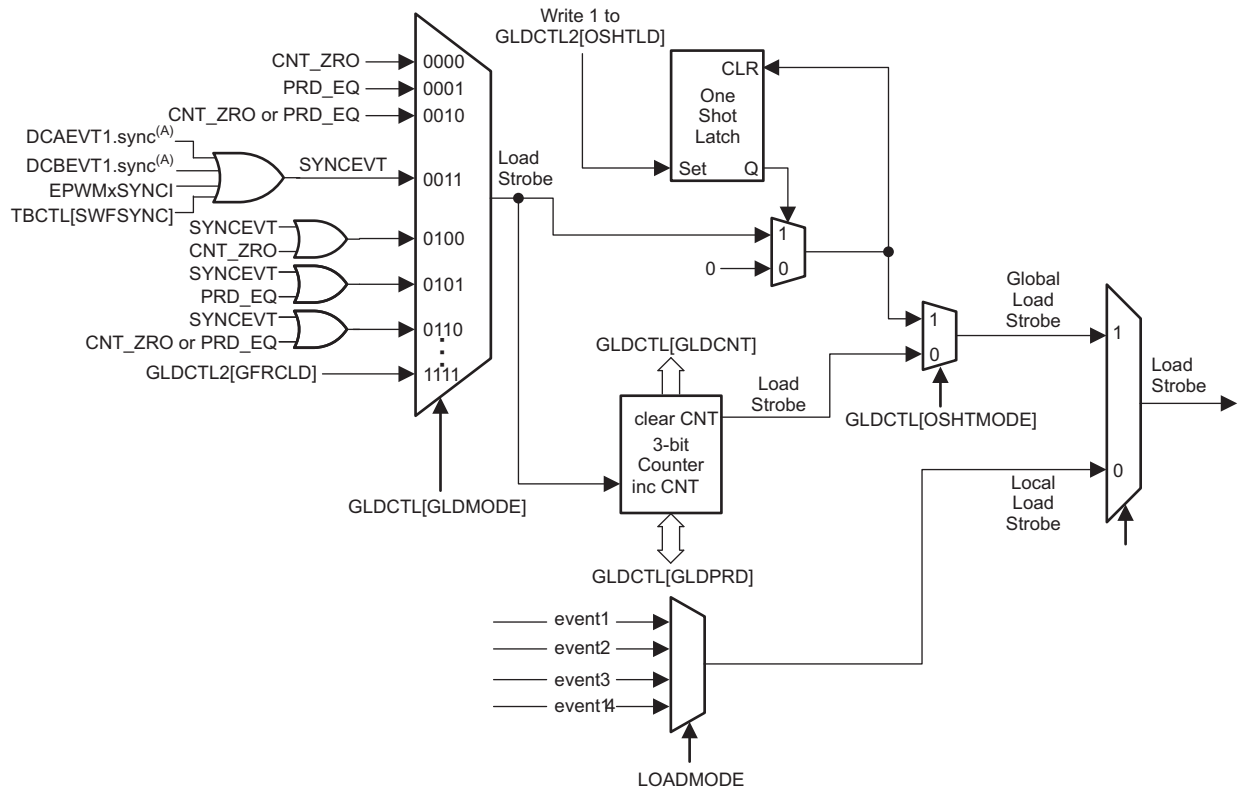


**Figure 18-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event**



### 18.4.7 Global Load

Figure 18-12 illustrates the signals and registers associated with the global load feature.

**Figure 18-12. Global Load: Signals and Registers**


When this feature is enabled, the transfer of contents from the shadow register to the active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in Global Shadow to Active Load Control Register (GLDCTL[GLDMODE]). When GLDCTL[GLD] = '1', shadow to active load event selection bits for individual shadowed registers are ignored and global load mode takes effect for the corresponding registers enabled by GLDCFG[REGx].

When GLDCTL[GLD] = '1' and GLDCFG[REGx] = '0' global load mode does not affect the corresponding register (REGx). Shadow to active load event selection bits for individual shadowed registers decide how the transfer of contents from shadow register to active register takes place.

#### 18.4.7.1 Global Load Pulse Pre-Scalar

This feature provides the capability to choose shadow to active transfers to happen once in 'N' occurrences of selected global load pulse (GLDCTL[GLDMODE]). This pre-scale functionality is not available for registers that cannot or are not configured to use the global load mechanism (that is, GLDCTL[GLD] = '0' or GLDCFG[REGx] = '0').

#### 18.4.7.2 One-Shot Load Mode

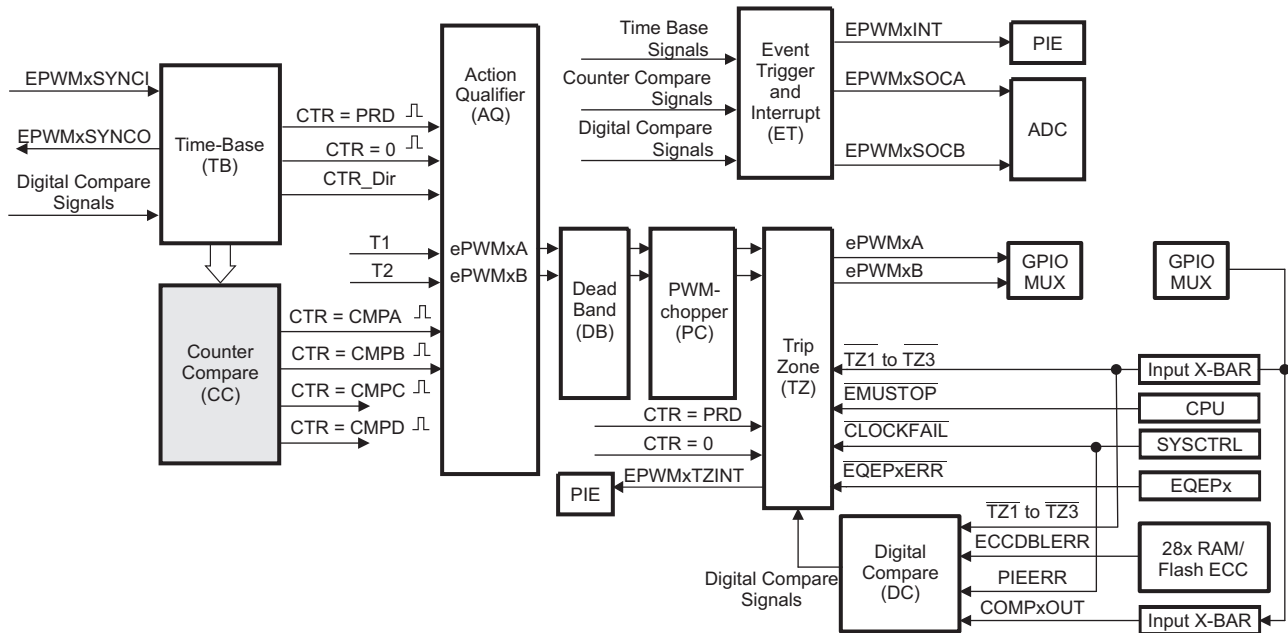
This feature allows users to cause the shadow register to active register transfers to occur once. When GLDCTL2[OSHTLD] = '1' the shadow to active register transfer, for registers that are configured to use the global load mechanism, takes place on the event selected by GLDCTL[GLDMODE].

Software force loading of contents from shadow register to active register is possible by using GLDCTL2[GFRCLD]. The GLDCTL2 register can also be linked across multiple PWM modules by using EPWMXLINK[GLDCTL2LINK]. This, along with the one-shot load mode feature discussed above, provides a method to correctly update multiple PWM registers in one or more PWM modules at certain PWM events or, if desired, in the same clock cycle. This is very useful in variable frequency applications and/or multi-phase interleaved applications.

## 18.5 Counter-Compare (CC) Submodule

Figure 18-13 illustrates the counter-compare submodule within the ePWM.

Figure 18-13. Counter-Compare Submodule



### 18.5.1 Purpose of the Counter-Compare Submodule

The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (CMPA) counter-compare B (CMPB) counter-compare C (CMPC) and counter-compare D (CMPD) registers. When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.

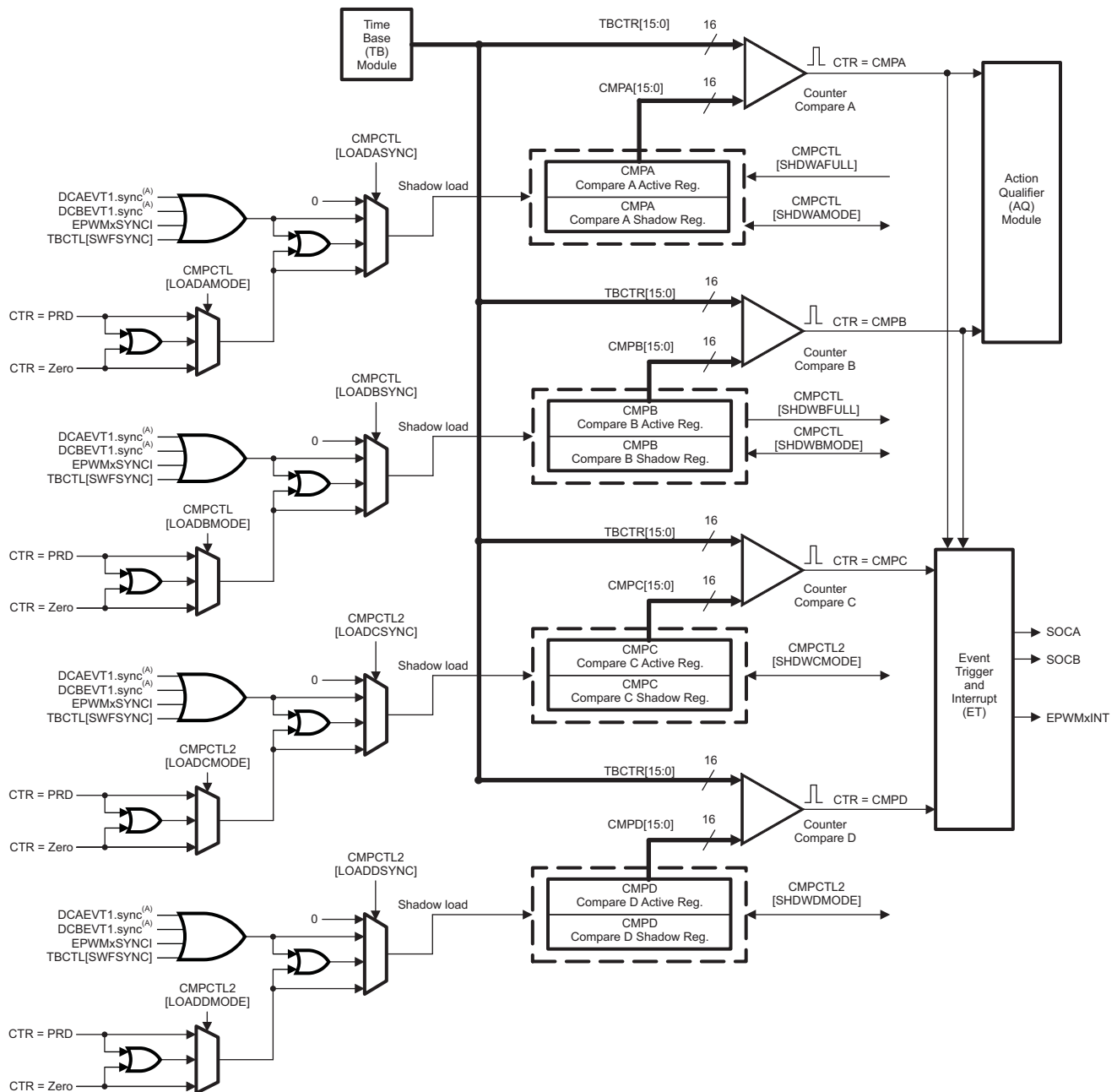
The counter-compare:

- Generates events based on programmable time stamps using the CMPA, CMPB, CMPC and CMPD registers
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCTR = CMPB)
  - CTR = CMPC: Time-base counter equals counter-compare C register (TBCTR = CMPC)
  - CTR = CMPD: Time-base counter equals counter-compare D register (TBCTR = CMPD)
- Controls the PWM duty cycle if the action-qualifier submodule is configured appropriately using counter-compare A (CMPA) & counter-compare B (CMPB)
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle

### 18.5.2 Controlling and Monitoring the Counter-Compare Submodule

The counter-compare submodule operation is shown in Figure 18-14.

Figure 18-14. Detailed View of the Counter-Compare Submodule



A These events are generated by the type 4 ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs (for example, CMPSSx and TZ signals).

### 18.5.3 Operational Highlights for the Counter-Compare Submodule

The counter-compare submodule is responsible for generating events which can be used in the action-qualifier and/or event-trigger submodules. There are four independent compare events described below:

1. CTR = CMPA: Time-base counter equal to counter-compare A register (TBCTR = CMPA).
2. CTR = CMPB: Time-base counter equal to counter-compare B register (TBCTR = CMPB).
3. CTR = CMPC: Time-base counter equal to counter-compare C register (TBCTR = CMPC). This event can be used to generate an event in the event trigger submodule only.
4. CTR = CMPD: Time-base counter equal to counter-compare D register (TBCTR = CMPD). This event



can be used to generate an event in the event trigger submodule only

For up-count or down-count mode, each event occurs only once per cycle. For up-down count mode each event occurs twice per cycle if the compare value is between 0x00-TBPRD and once per cycle if the compare value is equal to 0x00 or equal to TBPRD. These events are fed into the action-qualifier submodule where they are qualified by the counter direction and converted into actions if enabled. Refer to [Section 18.6.1](#) for more details.

The counter-compare registers CMPA and CMPB each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occur at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. Which register is written to or read from is determined by the CMPCTL[SHDWAMODE] and CMPCTL[SHDWBMODE] bits. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is described below:

#### Shadow Mode:

The shadow mode for the CMPA is enabled by clearing the CMPCTL[SHDWAMODE] bit and the shadow register for CMPB is enabled by clearing the CMPCTL[SHDWBMODE] bit. Shadow mode is enabled by default for both CMPA and CMPB.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the CMPCTL[LOADAMODE] CMPCTL[LOADBMODE] CMPCTL[LOADASYNC] & CMPCTL[LOADBSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or EPWMxSYNCl or TBCTL[SWFSYNC]
- Both SYNC event or a selection made by LOADAMODE/LOADBMODE

Only the active register contents are used by the counter-compare submodule to generate events to be sent to the action-qualifier.

#### Immediate Mode:

If immediate load mode is selected (that is, CMPCTL[SHDWAMODE] = 1 or CMPCTL[SHDWBMODE] = 1), then a read from or a write to the register will go directly to the active register.

#### Additional Comparators

The counter-compare submodule on ePWMs type 2 and later are responsible for generating two additional independent compare events based on two compare registers, which is fed to Event Trigger submodule :

1. CTR = CMPC: Time-base counter equal to counter-compare C register (TBCTR = CMPC).
2. CTR = CMPD: Time-base counter equal to counter-compare D register (TBCTR = CMPD).

The counter-compare registers CMPC and CMPD each have an associated shadow register. By default this register is shadowed. The memory address of the active register and the shadow register is identical. The value in the active CMPC and CMPD register is compared to the time-base counter (TBCTR). When the values are equal, the counter compare module generates a “time-base counter equal to counter compare C or counter compare D ” event respectively. Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWCMODE] and CMPCTL2[SHDWDMODE] bit. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is described below:

#### Shadow Mode:

The shadow mode for the CMPC is enabled by clearing the CMPCTL2[SHDWCMODE] bit and the shadow register for CMPD is enabled by clearing the CMPCTL2[SHDWDMODE] bit. Shadow mode is enabled by default for both CMPC and CMPD.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the CMPCTL2[LOADCMODE] CMPCTL2[LOADDDMODE] CMPCTL2[LOADCSYNC] & CMPCTL2[LOADDSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or EPWMxSYNCl or TBCTL[SWFSYNC]
- Both SYNC event or a selection made by LOADCMODE/LOADDDMODE

Only the active register contents are used by the counter-compare submodule to generate events to be sent to the action-qualifier.

#### **Immediate Load Mode:**

If the immediate load mode is selected (that is, CMPCTL2[SHDWCMODE] = 1 or CMPCTL2[SHDWDMODE] = 1), then a read from or a write to the register will go directly to the active register.

#### **Global Load Support**

The global load control mechanism can also be used for all counter-compare registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When the global load mode is selected the transfer of contents from shadow register to active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in the Global Shadow to Active Load Control Register (GLDCTL). The global load control mechanism is explained in [Section 18.4.7](#).

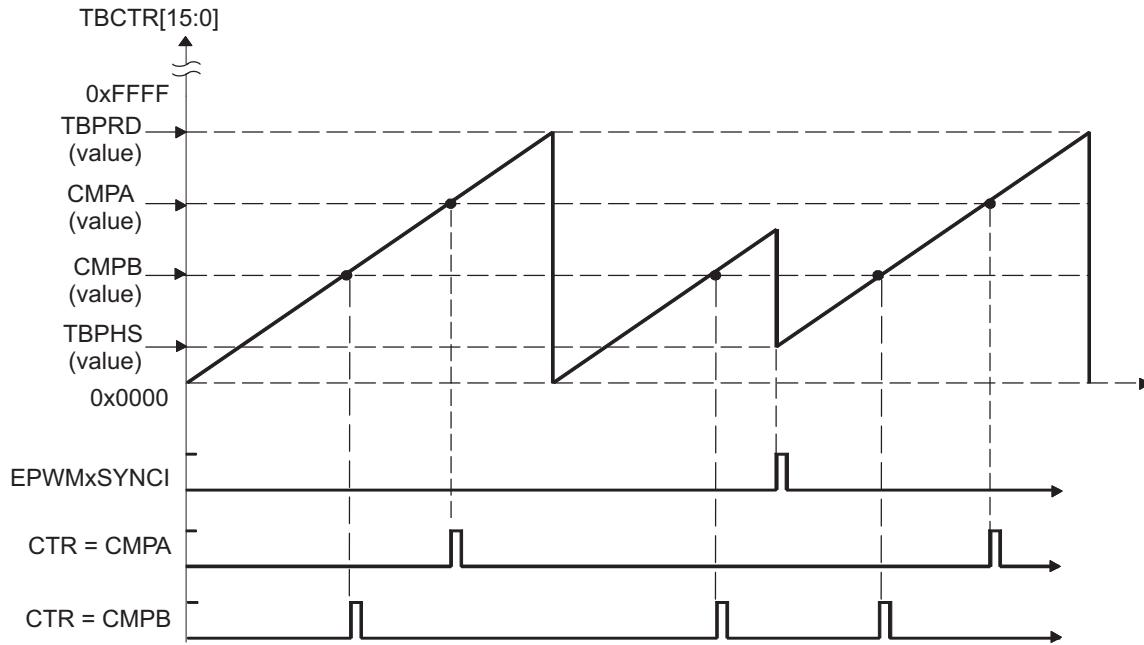
### **18.5.4 Count Mode Timing Waveforms**

The counter-compare module can generate compare events in all three count modes:

- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.
- Up-down-count mode: used to generate a symmetrical PWM waveform.

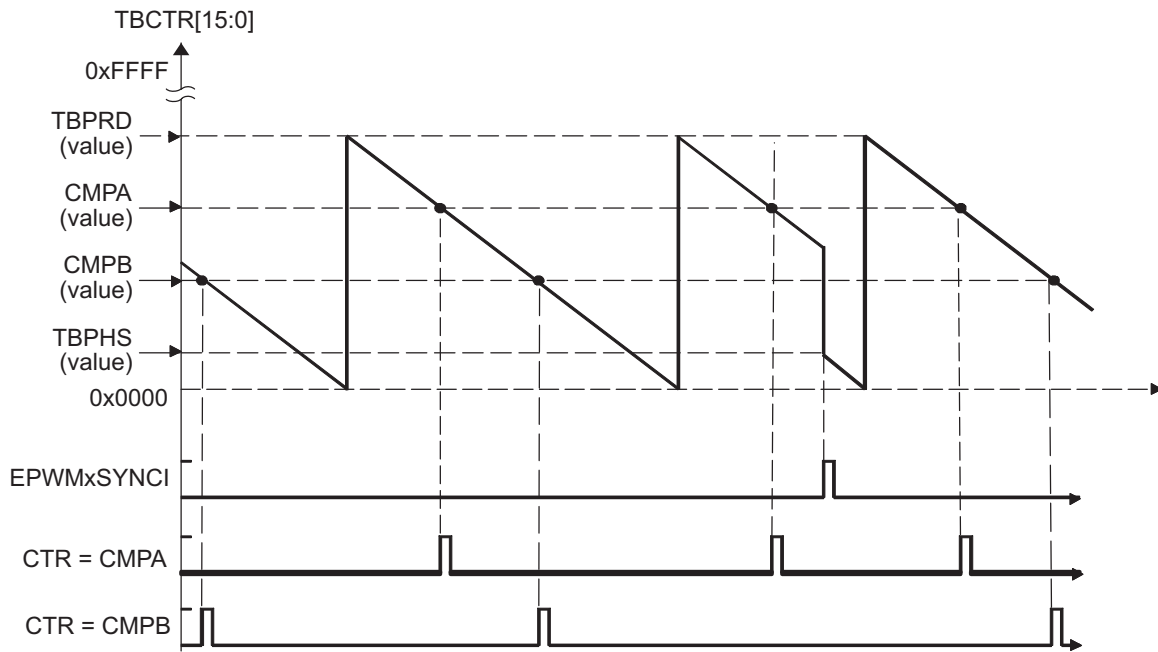
To best illustrate the operation of the first three modes, the timing diagrams in [Figure 18-15](#) through [Figure 18-18](#) show when events are generated and how the EPWMxSYNCl signal interacts.

**Figure 18-15. Counter-Compare Event Waveforms in Up-Count Mode**

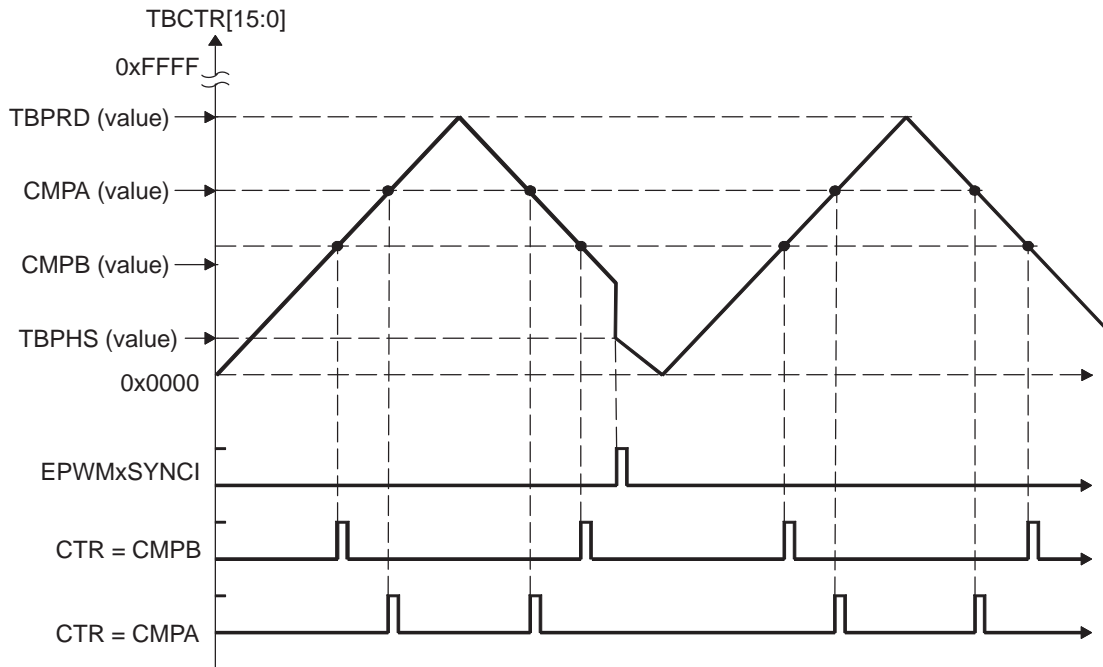


NOTE: An EPWMxSYNCl external synchronization event can cause a discontinuity in the TBCTR count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

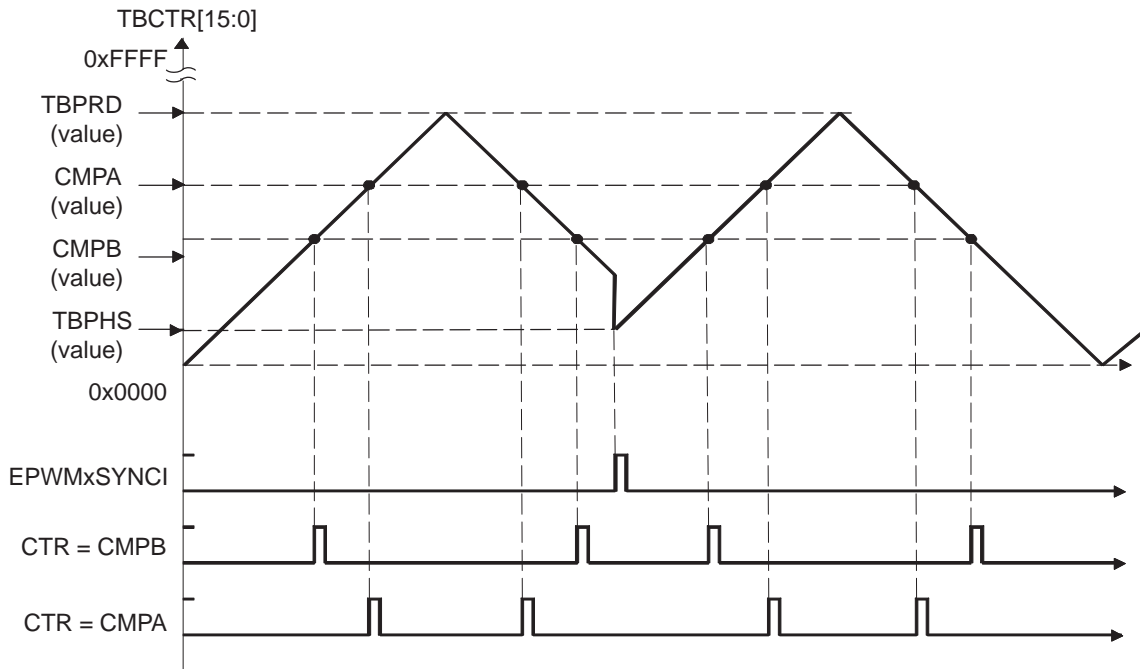
Figure 18-16. Counter-Compare Events in Down-Count Mode



**Figure 18-17. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event**

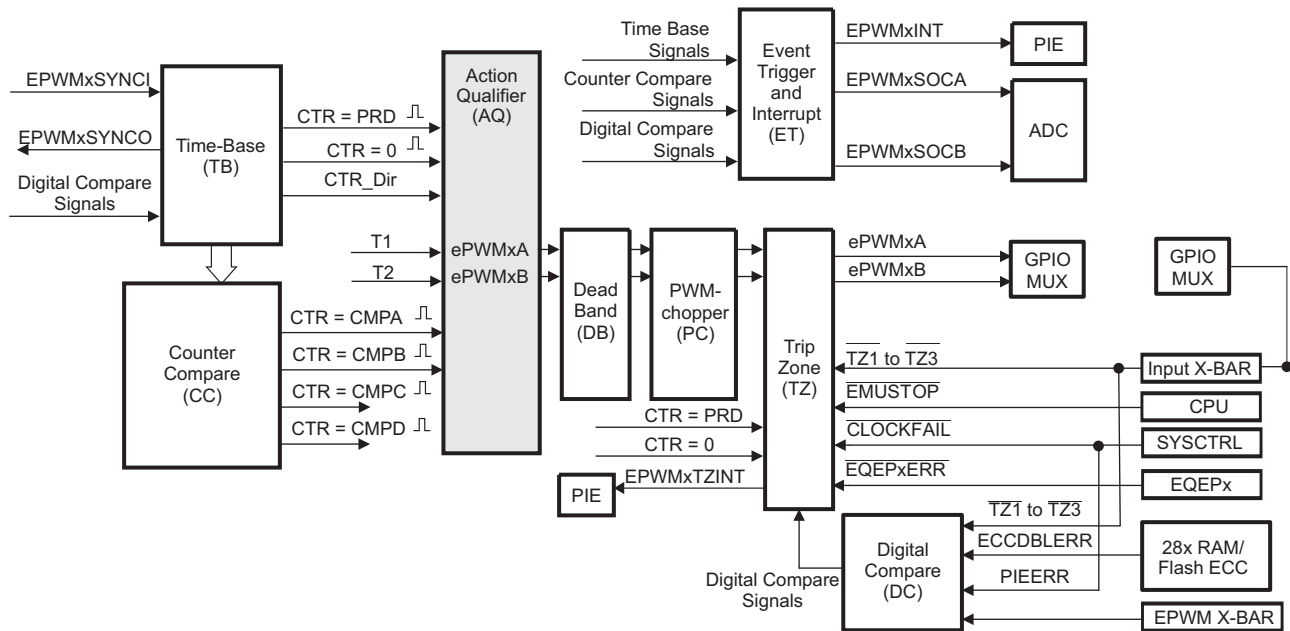


**Figure 18-18. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event**



## 18.6 Action-Qualifier (AQ) Submodule

The figure below shows the action-qualifier (AQ) submodule in the ePWM system.

**Figure 18-19. Action-Qualifier Submodule**


The action-qualifier submodule has the most important role in waveform construction and PWM generation. It decides which events are converted into various action types, thereby producing the required switched waveforms at the EPWMxA and EPWMxB outputs.

### 18.6.1 Purpose of the Action-Qualifier Submodule

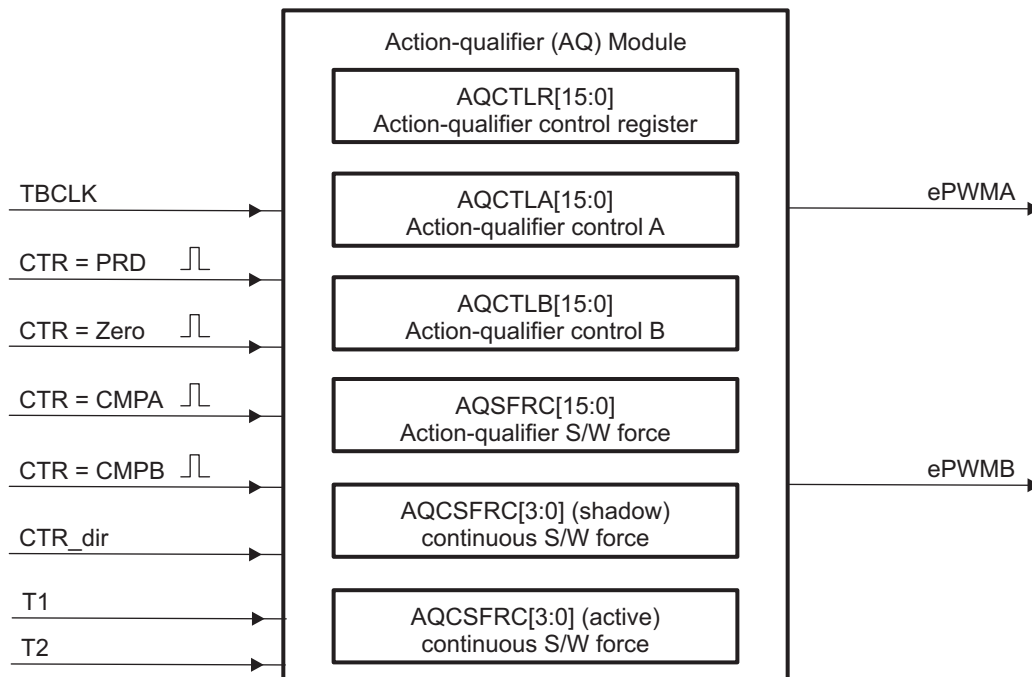
The action-qualifier submodule is responsible for the following:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
  - CTR = CMPA: Time-base counter equal to the counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equal to the counter-compare B register (TBCTR = CMPB)
- T1, T2 events: Trigger events based on comparator, trip or syncin events
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing

### 18.6.2 Action-Qualifier Submodule Control and Status Register Definitions

The action-qualifier submodule operation is shown in the figure below and monitored via the registers in [Section 18.16](#).

Figure 18-20. Action-Qualifier Submodule Inputs and Outputs



For convenience, the possible input events are summarized again in the table below.

Table 18-3. Action-Qualifier Submodule Possible Input Events

Signal	Description	Registers Compared
CTR = PRD	Time-base counter equal to the period value	TBCTR = TBPRD
CTR = Zero	Time-base counter equal to zero	TBCTR = 0x00
CTR = CMPA	Time-base counter equal to the counter-compare A	TBCTR = CMPA
CTR = CMPB	Time-base counter equal to the counter-compare B	TBCTR = CMPB
T1 event	Based on comparator, trip or syncin events	None
T2 event	Based on comparator, trip or syncin events	None
Software forced event	Asynchronous event initiated by software	

The software forced action is a useful asynchronous event. This control is handled by the AQSFR and AQCSFRC registers.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

The possible actions imposed on outputs EPWMxA and EPWMxB are:





























- **Set High:**  
Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:**  
Set output EPWMxA or EPWMxB to a low level.
- **Toggle:**  
If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:**  
Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option

prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts and ADC start of conversion. See the description in the [Section 18.10](#) for details.

Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. For example, both CTR = CMPA and CTR = CMPB can operate on output EPWMxA. All qualifier actions are configured via the control registers found at the end of this section.

For clarity, the drawings in this document use a set of symbolic actions. These symbols are summarized in [Figure 18-21](#). Each symbol represents an action as a marker in time. Some actions are fixed in time (zero and period) while the CMPA and CMPB actions are moveable and their time positions are programmed via the counter-compare A and B registers, respectively. To turn off or disable an action, use the "Do Nothing option"; it is the default at reset.

**Figure 18-21. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs**

SW force	TB Counter equals				Trigger Events		Actions
	Zero	Comp A	Comp B	Period	T1	T2	
							Do Nothing
							Clear Lo
							Set Hi
							Toggle

The Action Qualifier Trigger Event Source Selection register (AQTSRCSEL) is used to select the source for T1 and T2 events. T1/T2 selection and configuration of a trip/digital-compare event in Action Qualifier submodule is independent of the configuration of that event in the Trip-Zone submodule. A particular trip event may or may not be configured to cause trip action in the Trip Zone submodule, but the same event can be used by the Action Qualifier to generate T1/T2 for controlling PWM generation.



### 18.6.3 Action-Qualifier Event Priority

It is possible for the ePWM action qualifier to receive more than one event at the same time. In this case events are assigned a priority by the hardware. The general rule is events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down-count mode are shown in the table below. A priority level of 1 is the highest priority and level 10 is the lowest. The priority changes slightly depending on the direction of TBCTR.

**Table 18-4. Action-Qualifier Event Priority for Up-Down-Count Mode**

Priority Level	Event If TBCTR is Incrementing TBCTR = Zero up to TBCTR = TBPRD	Event If TBCTR is Decrementing TBCTR = TBPRD down to TBCTR = 1
1 (Highest)	Software forced event	Software forced event
2	T1 on up-count (T1U)	T1 on down-count (T1D)
3	T2 on up-count (T2U)	T2 on down-count (T2D)
4	Counter equals CMPB on up-count (CBU)	Counter equals CMPB on down-count (CBD)
5	Counter equals CMPA on up-count (CAU)	Counter equals CMPA on down-count (CAD)
6	Counter equals zero	Counter equals period (TBPRD)
7	T1 on down-count (T1D)	T1 on up-count (T1U)
8	T2 on down-count (T2D)	T2 on up-count (T2U)
9	Counter equals CMPB on down-count (CBD)	Counter equals CMPB on up-count (CBU)
10 (Lowest)	Counter equals CMPA on down-count (CAD)	Counter equals CMPA on up-count (CAU)

The table below shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up and therefore, down-count events will never be taken.

**Table 18-5. Action-Qualifier Event Priority for Up-Count Mode**

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to period (TBPRD)
3	T1 on up-count (T1U)
4	T2 on up-count (T2U)
5	Counter equal to CMPB on up-count (CBU)
6	Counter equal to CMPA on up-count (CAU)
7 (Lowest)	Counter equal to Zero

The table below shows the action-qualifier priority for down-count mode. In this case, the counter direction is always defined as down and thus up-count events will never be taken.

**Table 18-6. Action-Qualifier Event Priority for Down-Count Mode**

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to Zero
3	T1 on down-count (T1D)
4	T2 on down-count (T2D)
5	Counter equal to CMPB on down-count (CBD)
6	Counter equal to CMPA on down-count (CAD)
7 (Lowest)	Counter equal to period (TBPRD)

It is possible to set the compare value greater than the period. In this case the action will take place as shown in the table below.

**Table 18-7. Behavior if CMPA/CMPB is Greater than the Period**

Counter Mode	Compare on Up-Count Event CAD/CBD	Compare on Down-Count Event CAD/CBD
Up-Count Mode	If $CMPA/CMPB \leq TBPRD$ period, then the event occurs on a compare match (TBCTR=CMPA or CMPB).  If $CMPA/CMPB > TBPRD$ , then the event will not occur.	Never occurs.
Down-Count Mode	Never occurs.	If $CMPA/CMPB < TBPRD$ , the event will occur on a compare match (TBCTR=CMPA or CMPB).  If $CMPA/CMPB \geq TBPRD$ , the event will occur on a period match (TBCTR=TBPRD).
Up-Down-Count Mode	If $CMPA/CMPB < TBPRD$ and the counter is incrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB).  If $CMPA/CMPB \geq TBPRD$ , the event will occur on a period match (TBCTR = TBPRD).	If $CMPA/CMPB < TBPRD$ and the counter is decrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB).  If $CMPA/CMPB \geq TBPRD$ , the event occurs on a period match (TBCTR=TBPRD).

#### 18.6.4 AQCTLA and AQCTLB Shadow Mode Operations

To enable Action Qualifier mode changes which must occur at the end of a period even when the phase changes, shadowing of the AQCTLA and AQCTLB registers has been added on ePWMs type 2 and later. Additionally, shadow to active load on SYNC of these registers is supported as well. Shadowing of this register is enabled and disabled by the AQCTL[SHDWAQAMODE] and AQCTL[SHDWAQBMODE] bits. These bits enable and disable the AQCTLA shadow register and AQCTLB shadow register, respectively. The behavior of the two load modes is described below:

##### Shadow Mode:

The shadow mode for the AQCTLA is enabled by setting the AQCTL[SHDWAQAMODE] bit, and the shadow register for AQCTLB is enabled by setting the AQCTL[SHDWAQBMODE] bit. Shadow mode is disabled by default for both AQCTLA and AQCTLB

If the shadow register is enabled, then the content of the shadow register is transferred to the active register on one of the following events as specified by the AQCTL[LDAQAMODE] AQCTL[LDAQBMODE] AQCTL[LDAQASYNC] & AQCTL[LDAQBSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or EPWMxSYNCl or TBCTL[SWFSYNC]
- Both SYNC event or a selection made by LDAQAMODE/LDAQBMODE

##### Global Load Support

Global load control mechanism can also be used for AQCTLA:AQCTLA2, AQCTLB:AQCTLB2 and AQCSFRC registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When global load mode is selected, the transfer of contents from shadow register to active register for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in the Global Shadow to Active Load Control Register (GLDCTL). The global load control mechanism is explained in [Section 18.4.7](#).

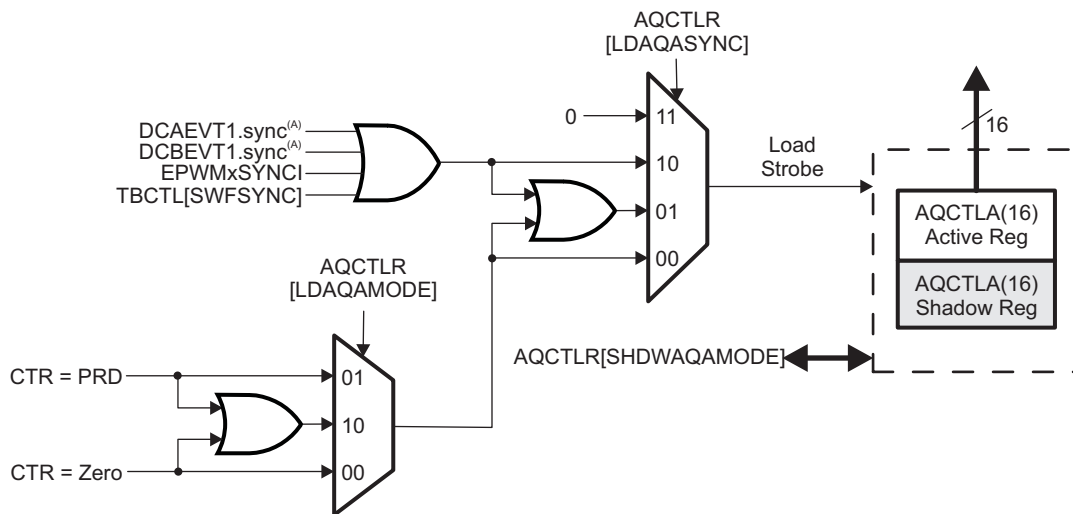
##### Immediate Load Mode:

If immediate load mode is selected (that is, AQCTL[SHDWAQAMODE] = 0 or AQCTL[SHDWAQBMODE] = 0), then a read from or a write to the register will go directly to the active register. See [Figure 18-22](#) and [Figure 18-23](#).

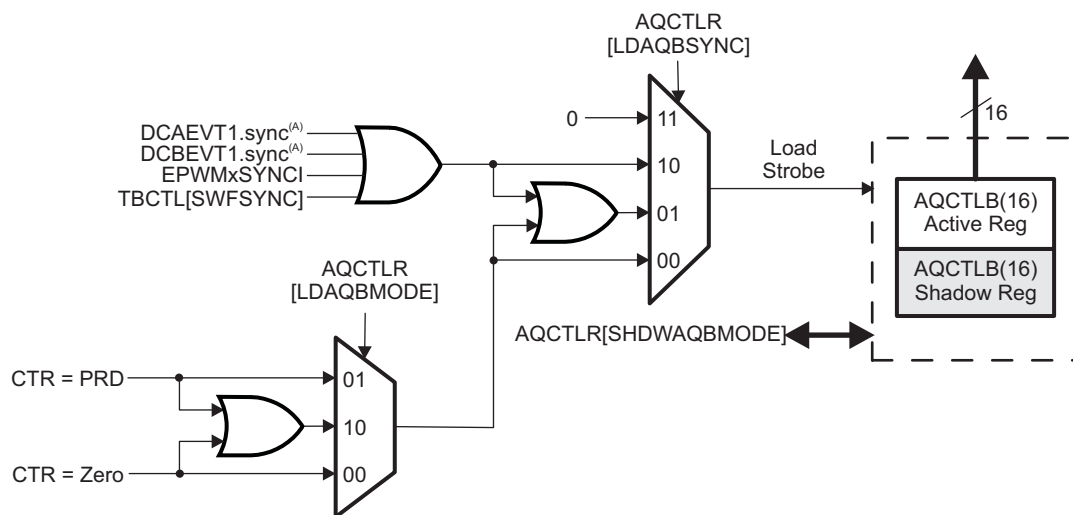
**NOTE:** Shadow to Active Load of Action Qualifier Output A/B Control Register [AQCTLA & AQCTLB] on CMPA = 0 or CMPB = 0 boundary

If the Counter-Compare A Register (CMPA) or Counter-Compare B Register (CMPB) is set to a value of 0 and the action qualifier action on AQCTLA and AQCTLB is configured to occur in the same instant as a shadow to active load (that is, CMPA=0 and AQCTLA shadow to active load on TBCTR=0 using AQCTL register LDAQAMODE and LDAQAMODE bits), then both events enter contention and it is recommended to use a Non-Zero Counter-Compare when using Shadow to Active Load of Action Qualifier Output A/B Control Register on TBCTR = 0 boundary.

**Figure 18-22. AQCTL[SHDWAQAMODE]**



**Figure 18-23. AQCTL[SHDWAQBMODE]**



## 18.6.5 Waveforms for Common Configurations

**NOTE:** The waveforms in this document show the behavior of the ePWMs for a static compare register value. In a running system, the active compare registers (CMPA and CMPB) are typically updated from their respective shadow registers once every period. The user specifies when the update will take place; either when the time-base counter reaches zero or when the time-base counter reaches period. There are some cases when the action based on the new value can be delayed by one period or the action based on the old value can take effect for an extra period. Some PWM configurations avoid this situation. These include, but are not limited to, the following:

### Use up-down-count mode to generate a symmetric PWM:

- If you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1.
- If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1.

This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

### Use up-down-count mode to generate an asymmetric PWM:

- To achieve 50%-0% asymmetric PWM use the following configuration: Load CMPA/CMPB on period and use the period action to clear the PWM and a compare-up action to set the PWM. Modulate the compare value from 0 to TBPRD to achieve 50%-0% PWM duty.

### When using up-count mode to generate an asymmetric PWM:

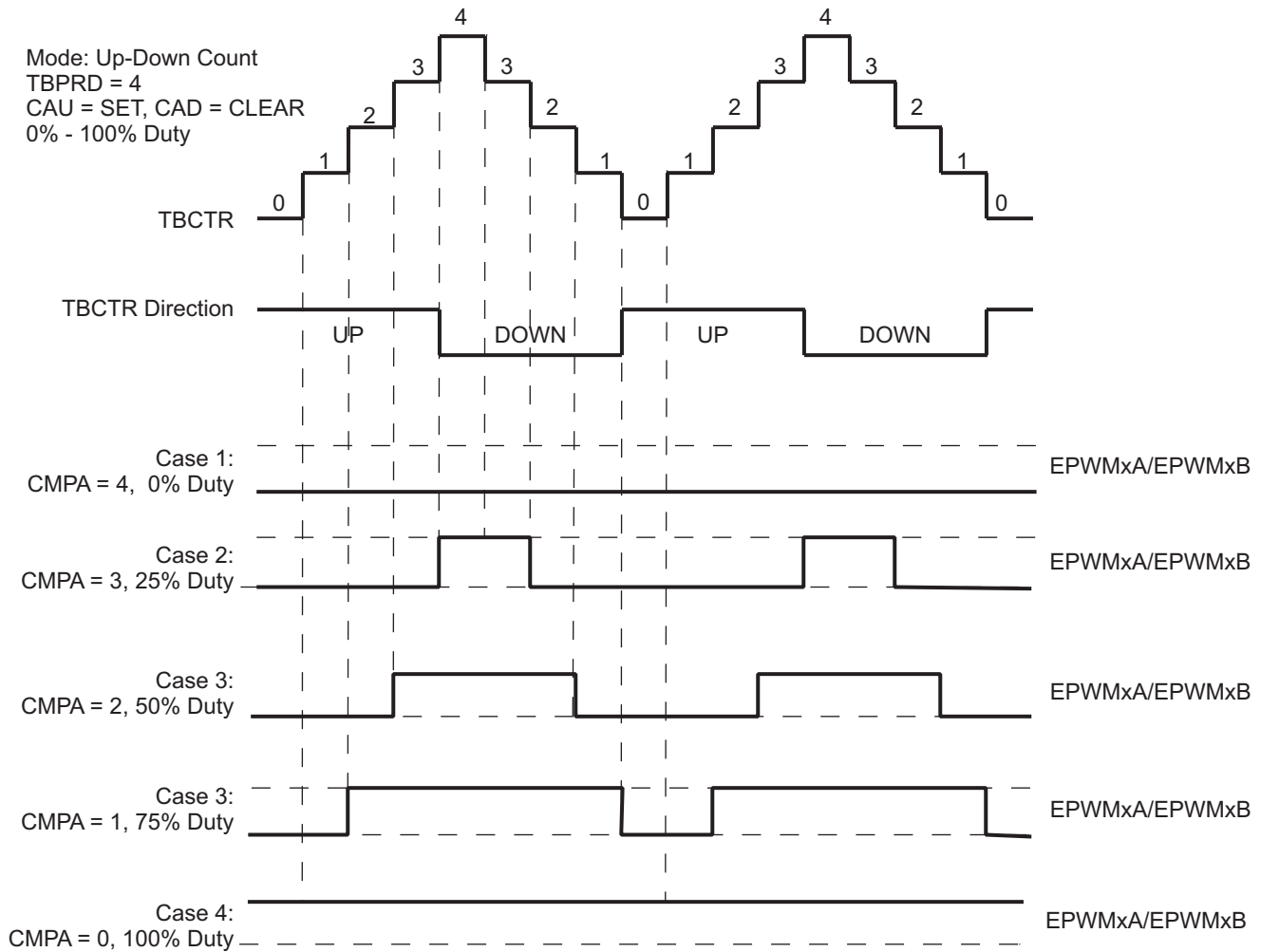
- To achieve 0-100% asymmetric PWM use the following configuration: Load CMPA/CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to TBPRD+1 to achieve 0-100% PWM duty.

See the *Using Enhanced Pulse Width Modulator (ePWM) Module for 0-100% Duty Cycle Control* Application Report (literature number [SPRAA11](#))

The figure below shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCTR. In this mode 0%-100% DC modulation is achieved by using equal compare matches on the up count and down count portions of the waveform. In the example shown, CMPA is used to make the comparison. When the counter is incrementing the CMPA match will pull the PWM output high. Likewise, when the counter is decrementing the compare match will pull the PWM signal low. When  $CMPA = 0$ , the PWM signal is low for the entire period giving the 0% duty waveform. When  $CMPA = TBPRD$ , the PWM signal is high achieving 100% duty.

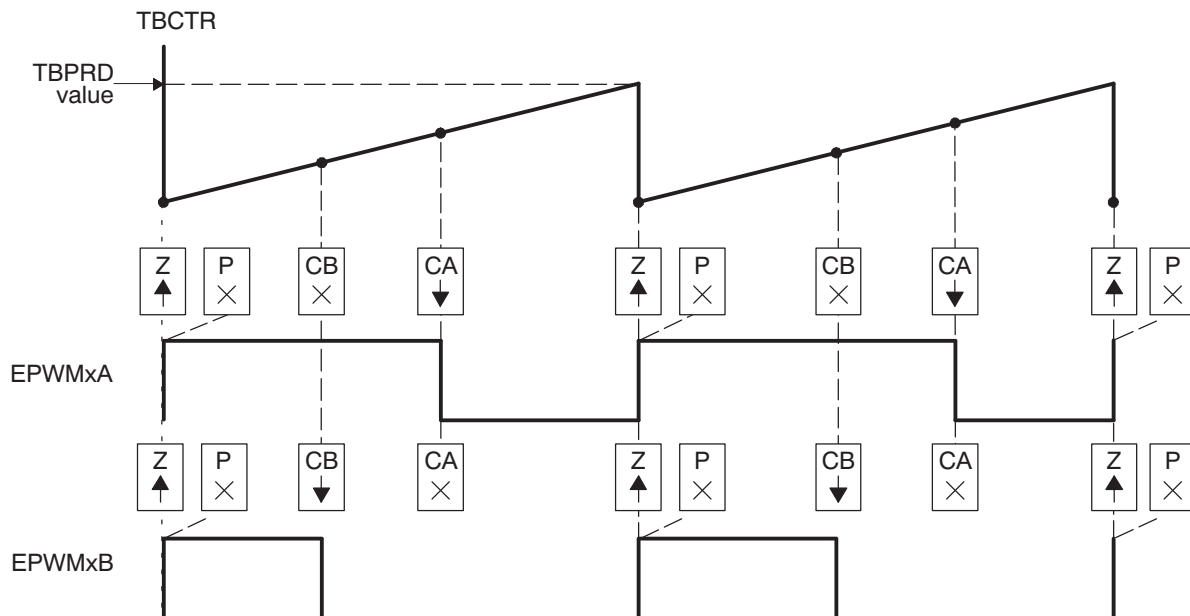
When using this configuration in practice, if you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1. If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1. This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

**Figure 18-24. Up-Down-Count Mode Symmetrical Waveform**



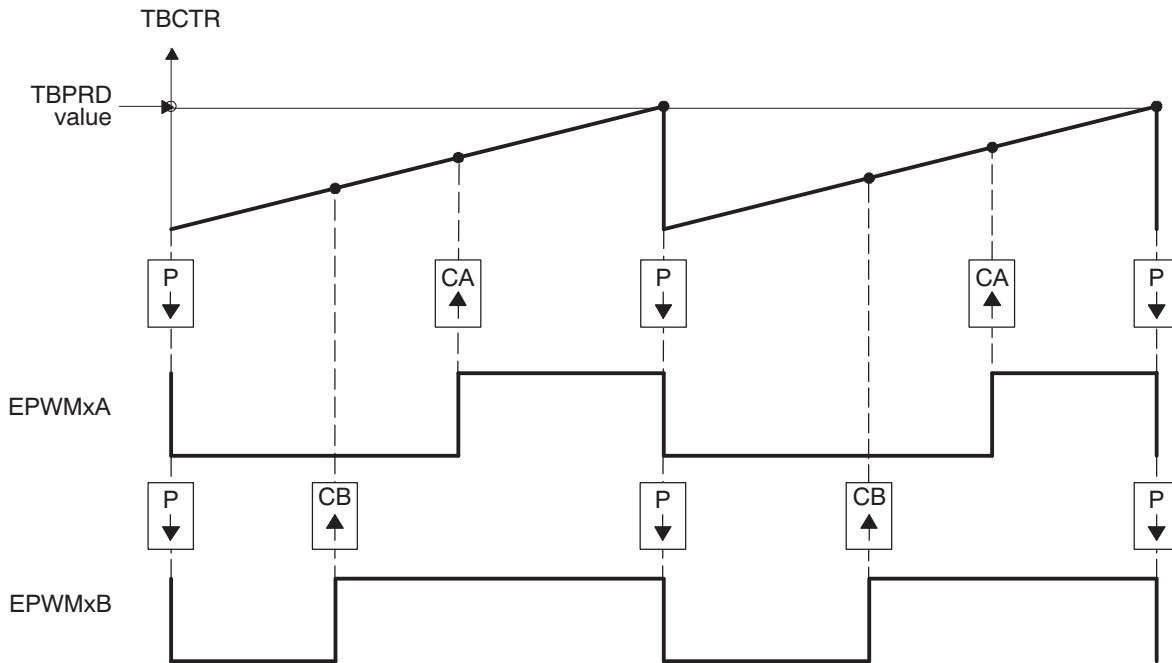
The PWM waveforms in [Figure 18-25](#) through [Figure 18-30](#) show some common action-qualifier configurations. Some conventions used in the figures and examples are as follows:

- TBPRD, CMPA, and CMPB refer to the value written in their respective registers. The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB.
- EPWMxA and EPWMxB refer to the output signals from ePWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

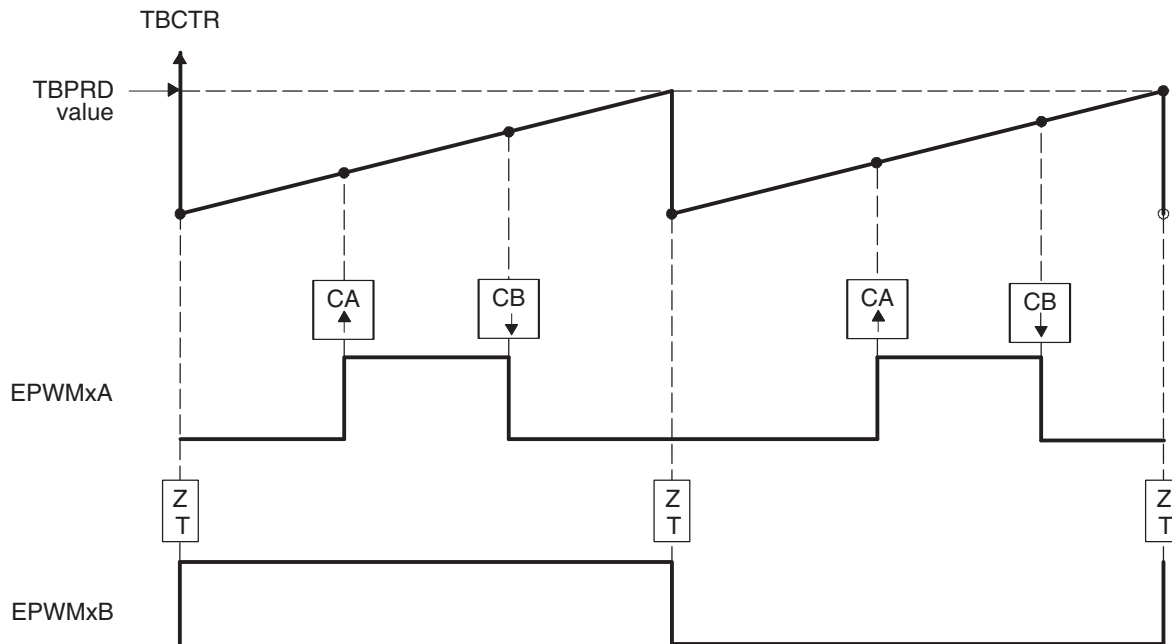
**Figure 18-25. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High**


- A PWM period =  $(TBPRD + 1) \times T_{TBCLK}$
- B Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).
- C Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).
- D The "Do Nothing" actions ( X ) are shown for completeness, but will not be shown on subsequent diagrams.
- E Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

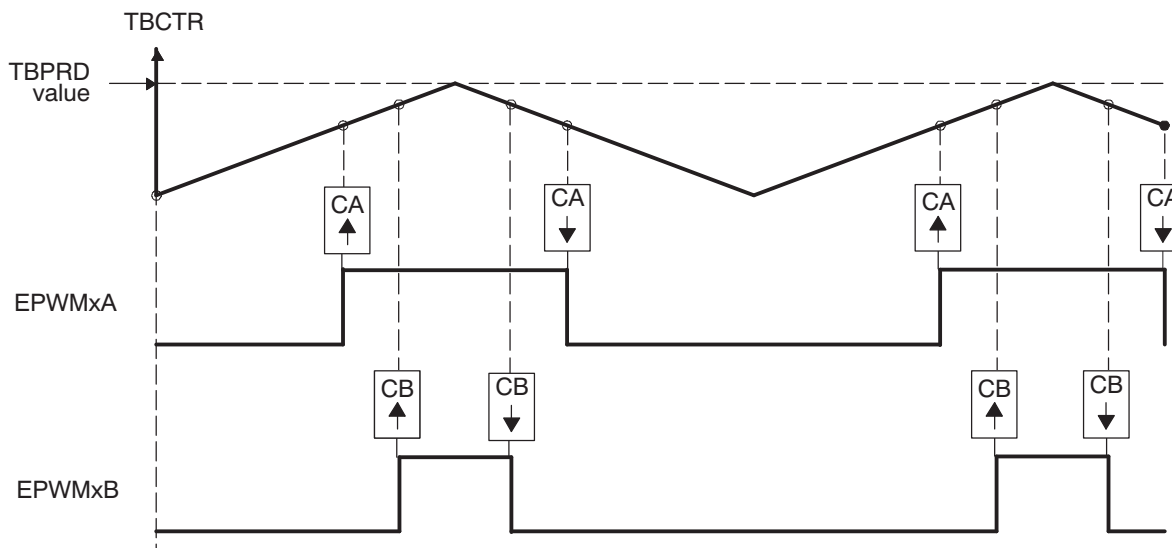
**Figure 18-26. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low**



- A PWM period =  $(TBPRD + 1) \times T_{TBCLK}$
- B Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- C Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- D Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

**Figure 18-27. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA**


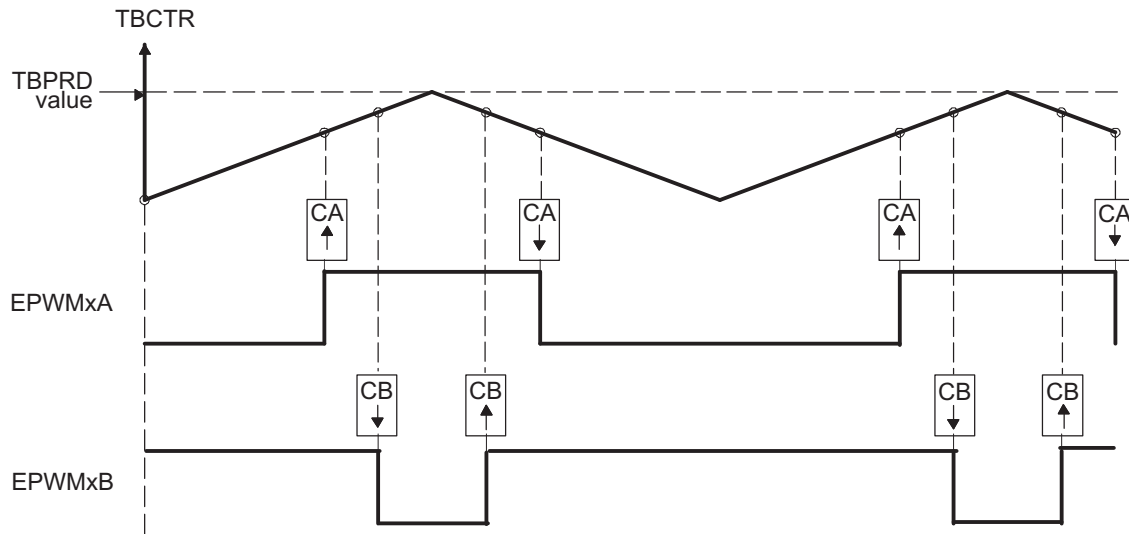
- A  $\text{PWM frequency} = 1 / ((\text{TBPRD} + 1) \times T_{\text{TBCLK}})$
- B Pulse can be placed anywhere within the PWM cycle (0000 - TBPRD)
- C High time duty proportional to (CMPB - CMPA)

**Figure 18-28. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low**


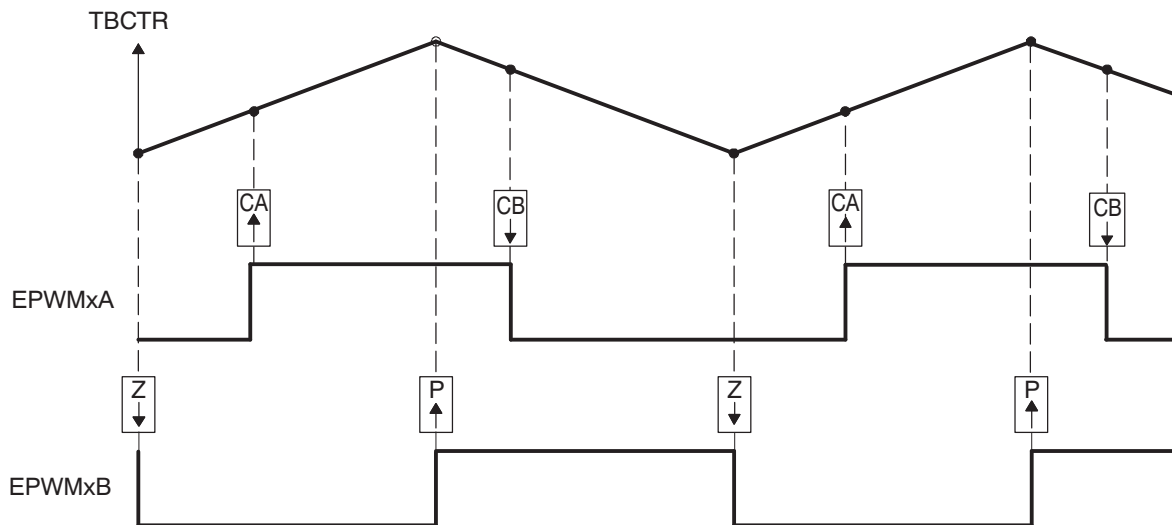
- A  $\text{PWM period} = 2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- B Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- C Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- D Outputs EPWMxA and EPWMxB can drive independent power switches.



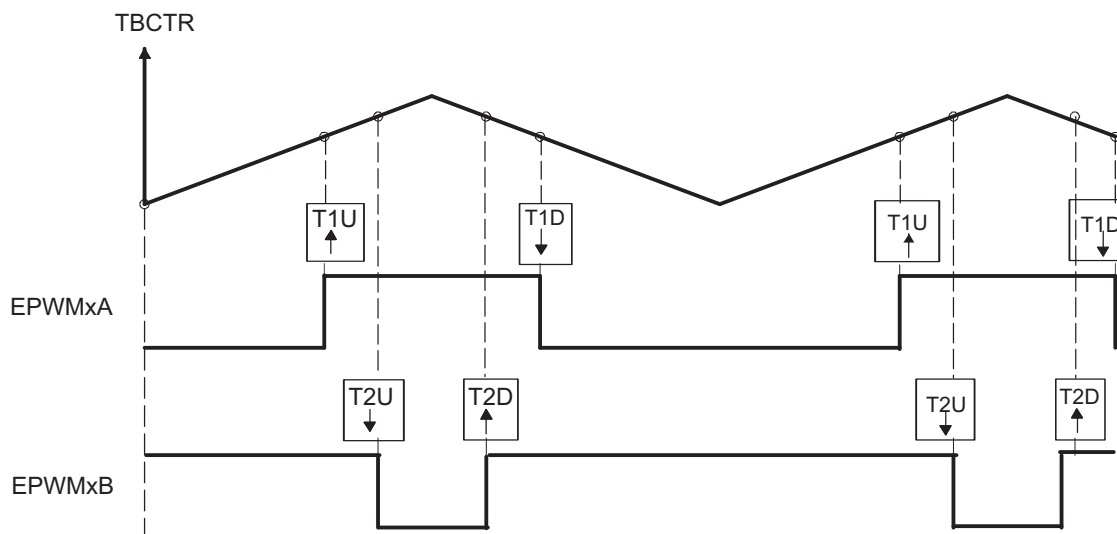
**Figure 18-29. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary**



- A PWM period =  $2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- B Duty modulation for EPWMxA is set by CMPA, and is active low, that is, low time duty proportional to CMPA.
- C Duty modulation for EPWMxB is set by CMPB and is active high, that is, high time duty proportional to CMPB.
- D Outputs EPWMx can drive upper/lower (complementary) power switches.
- E Dead-band =  $\text{CMPB} - \text{CMPA}$  (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

**Figure 18-30. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low**


- A PWM period =  $2 \times \text{TBPRD} \times \text{TBCLK}$
- B Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.
- C Duty modulation for EPWMxA is set by CMPA and CMPB.
- D Low time duty for EPWMxA is proportional to  $(\text{CMPA} + \text{CMPB})$ .
- E To change this example to active high, CMPA and CMPB actions need to be inverted (that is, Clear on CMPA, Set on CMPB).
- F Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB).

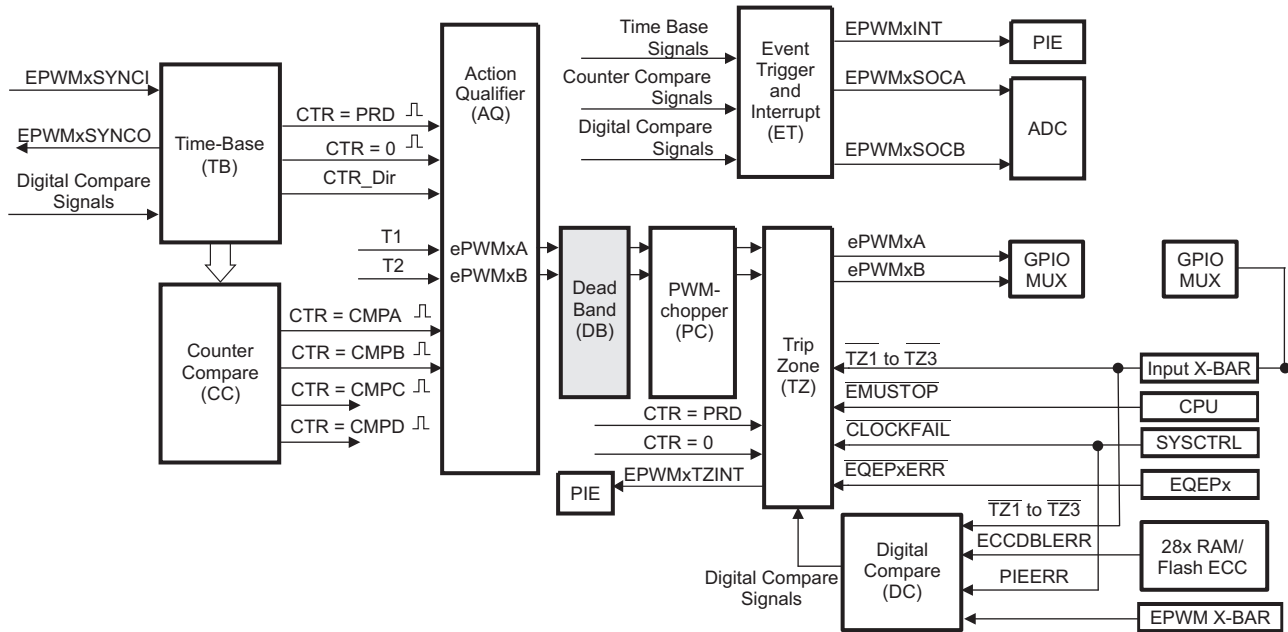
**Figure 18-31. Up-Down-Count, PWM Waveform Generation Utilizing T1 and T2 Events**


- A PWM period =  $2 \times \text{TBPRD} \times \text{TTBCLK}$
- B Independent T1 event actions when counter is counting up and when it is counting down are used to generate EPWMxA output.
- C Independent T2 event actions when counter is counting up and when it is counting down are used to generate EPWMxB output.
- D TZ1 is selected as the source for T1.
- E TZ2 is selected as the source for T2.

## 18.7 Dead-Band Generator (DB) Submodule

Figure 18-32 illustrates the dead-band submodule within the ePWM module.

Figure 18-32. Dead\_Band Submodule



### 18.7.1 Purpose of the Dead-Band Submodule

The action-qualifier (AQ) module section discussed how it is possible to generate the required dead band by having full control over edge placement using both the CMPA and CMPB resources of the ePWM module. However, if the more classical edge delay-based dead band with polarity control is required, then the dead-band submodule described here should be used.

The key functions of the dead-band module are:

- Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMxA input
- Programming signal pairs for:
  - Active high (AH)
  - Active low (AL)
  - Active high complementary (AHC)
  - Active low complementary (ALC)
- Adding programmable delay to rising edges (RED)
- Adding programmable delay to falling edges (FED)
- Can be totally bypassed from the signal path (note dotted lines in diagram)

### 18.7.2 Dead-band Submodule Additional Operating Modes

On type 1 ePWM RED could appear on one channel output and FED could appear on the other channel output.

The following list shows the distinct difference between type 1 and type 4 modules with respect to dead-band operating modes:

- By adding S6, S7, and S8 in Figure 18-33, RED and FED can appear on both the A-channel and B-channel outputs. Additionally, both RED and FED together can be applied to either the A-channel or B-channel outputs to allow B-channel phase shifting with respect to the A-channel.

**Note:** Phase shifting B-channel with respect to the A-channel using the dead-band submodule

additional operating modes has limitations with respect to the choice of RED and FED delay with respect to the operating duty cycle of the ePWMxA and ePWMxB outputs.

- The dead-band counters have also been increased to 14 bits
- Dead-band and dead-band High-resolution registers are now shadowed.
- High-resolution dead-band RED and FED have been enabled using the DBREDHR and DBFEDHR registers

---

**NOTE:** The PWM chopper will not be enabled when high-resolution dead band is enabled.

---

**NOTE:** High-resolution dead-band RED and FED requires Half-Cycle clocking mode (DBCTL[HALFCYCLE] = 1).

Cannot have both RED and FED together applied to both ePWMxA and ePWMxB. RED and FED together can be applied only to either OutA OR OutB.

---

**NOTE:** Phase shifting B-channel with respect to the A-channel: When PWMxB is derived from PWMxA using the DEDB\_MODE bit and by delaying rising edge and falling edge by the phase shift amount. When the duty cycle value on PWMxA is less than this phase shift amount, PWMxA's falling edge has precedence over the delayed rising edge for PWMxB. It is recommended to make sure the duty cycle value of the current waveform fed to the dead-band module is greater than the required phase shift amount.

---

**NOTE:** The Type 4 action qualifier and deadband outputs of the ePWM module are delayed by one TBCLK cycle in comparison to the Type 2 ePWM module, although the Type 4 behavior is the same as the Type 3 PWM. Both PWMA and PWMB signals are delayed under all circumstances.

---

### Shadow Mode:

The shadow mode for the DBRED is enabled by setting the DBCTL[SHDWDBREDDMODE] bit and the shadow register for DBFED is enabled by setting the DBCTL [SHDWDBFEDMODE] bit. Shadow mode is disabled by default for both DBRED and DBFED

If the shadow register is enabled, then the content of the shadow register is transferred to the active register on one of the following events as specified by the DBCTL [LOADREDMODE] & DBCTL [LOADFEDMODE] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero

The DBCTL register can be shadowed. The shadow mode for DBCTL is enabled by setting the DBCTL2[SHDWDBCTLMODE] bit. If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the DBCTL2[LOADDBCTLMODE] register bit:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD)
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero

### Global Load Support

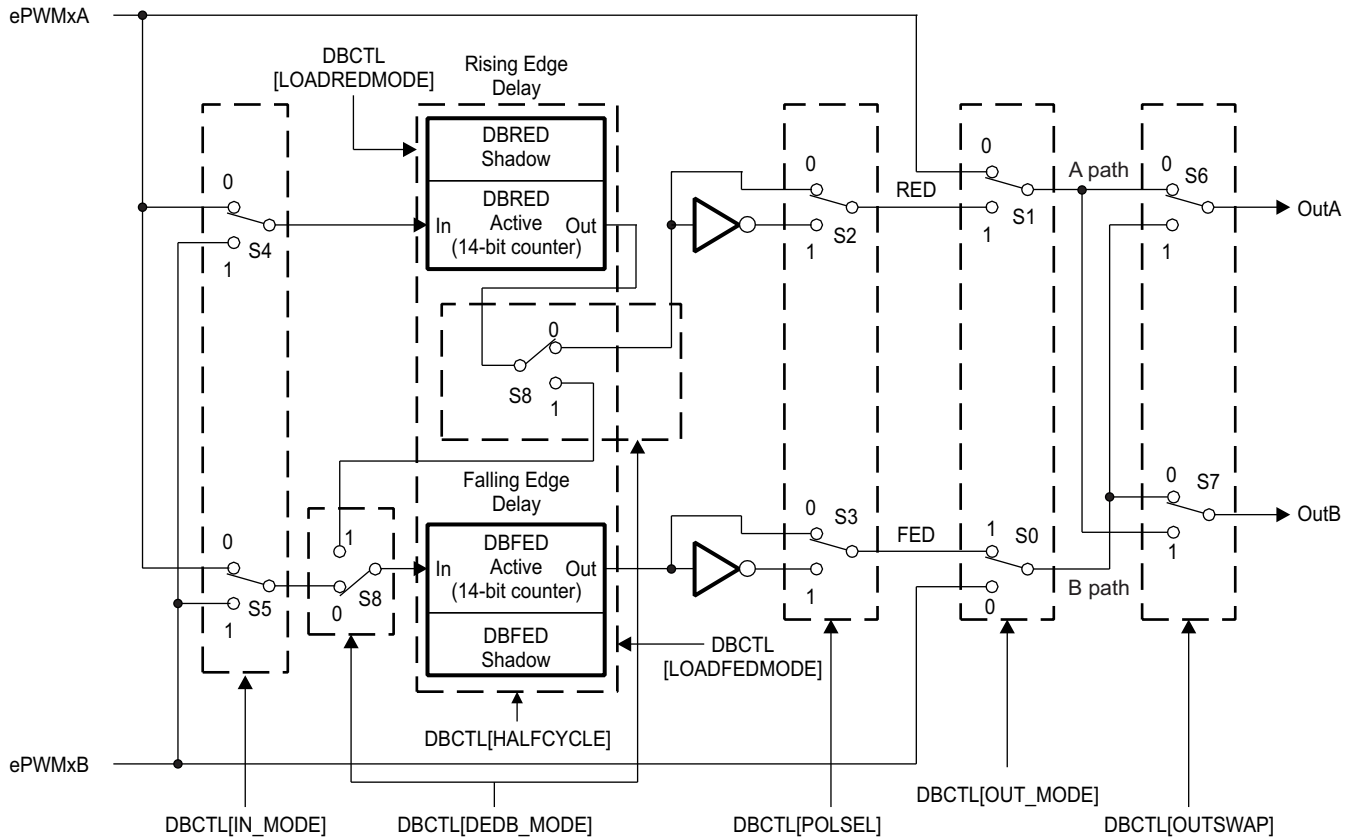
Global load control mechanism can also be used for DBRED:DBREDHR, DBFED:DBFEDHR and DBCTL registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When global load mode is selected the transfer of contents from shadow register to active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in the Global Shadow to Active Load Control Register (GLDCTL). The Global load control mechanism is explained in [Section 18.4.7](#).

**NOTE:** When DBRED/DBFED active is loaded with a new shadow value while DB counters are counting, the new DBRED / DBFED value only affects the NEXT PWMx edge and not the current edge.

### 18.7.3 Operational Highlights for the Dead-Band Submodule

The configuration options for the dead-band submodule are shown in [Figure 18-33](#).

**Figure 18-33. Configuration Options for the Dead-Band Submodule**



Although all combinations are supported, not all are typical usage modes. [Table 18-8](#) documents some classical dead-band configurations. These modes assume that the DBCTL[IN\_MODE] is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in [Table 18-8](#) fall into the following categories:

- **Mode 1: Bypass both falling-edge delay (FED) and rising-edge delay (RED)**  
Allows you to fully disable the dead-band submodule from the PWM signal path.
- **Mode 2-5: Classical Dead-Band Polarity Settings:**  
These represent typical polarity configurations that should address all the active high/low modes required by available industry power switch gate drivers. The waveforms for these typical cases are shown in [Figure 18-34](#). Note that to generate equivalent waveforms to [Figure 18-34](#), configure the action-qualifier submodule to generate the signal as shown for EPWMxA.
- **Mode 6: Bypass rising-edge-delay and Mode 7: Bypass falling-edge-delay**  
Finally the last two entries in [Table 18-8](#) show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

**Table 18-8. Classical Dead-Band Operating Modes**

Mode	Mode Description	DBCTL[POLSEL]		DBCTL[OUT_MODE]	
		S3	S2	S1	S0
1	EPWMxA and EPWMxB Passed Through (No Delay)	X	X	0	0
2	Active High Complementary (AHC)	1	0	1	1
3	Active Low Complementary (ALC)	0	1	1	1
4	Active High (AH)	0	0	1	1
5	Active Low (AL)	1	1	1	1
6	EPWMxA Out = EPWMxA In (No Delay)	0 or 1	0 or 1	0	1
	EPWMxB Out = EPWMxA In with Falling Edge Delay				
7	EPWMxA Out = EPWMxA In with Rising Edge Delay	0 or 1	0 or 1	1	0
	EPWMxB Out = EPWMxB In with No Delay				

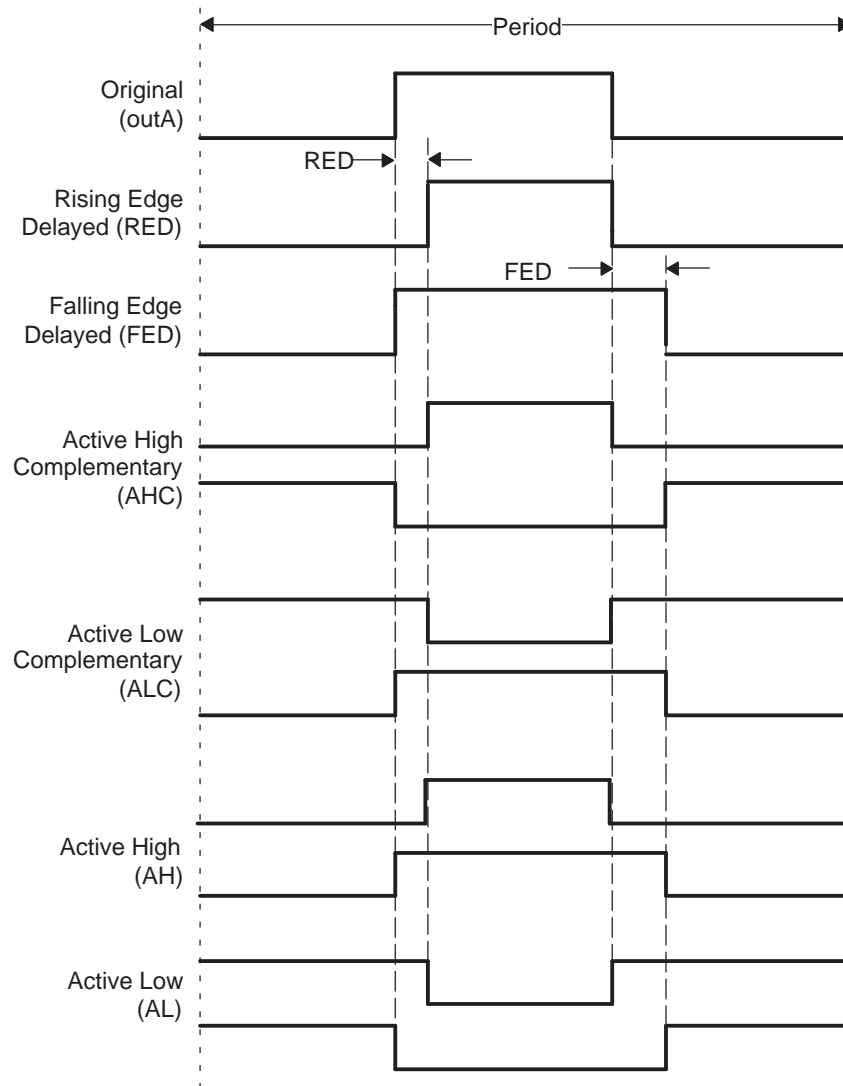
**Table 18-9. Additional Dead-Band Operating Modes**

Mode Description	DBCTL[DEDB-MODE]	DBCTL[OUTSWAP]	
	S8	S6	S7
EPWMxA and EPWMxB signals are as defined by OUT-MODE bits.	0	0	0
EPWMxA = A-path as defined by OUT-MODE bits.	0	0	1
EPWMxB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A-signal path)			
EPWMxA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B-signal path)	0	1	0
EPWMxB = B-path as defined by OUT-MODE bits			
EPWMxA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B-signal path)	0	1	1
EPWMxB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A-signal path)			
Rising edge delay applied to EPWMxA / EPWMxB as selected by S4 switch (IN-MODE bits) on A signal path only.	0	X	X
Falling edge delay applied to EPWMxA / EPWMxB as selected by S5 switch (IN-MODE bits) on B signal path only.			
Rising edge delay and falling edge delay applied to source selected by S4 switch (IN-MODE bits) and output to B signal path only. <sup>(1)</sup>	1	X	X

<sup>(1)</sup> When this bit is set to 1, user should always either set OUT\_MODE bits such that Apath = InA **or** OUTSWAP bits such that EPWMxA=Bpath. Otherwise, EPWMxA will be invalid.

Figure 18-34 shows waveforms for typical cases where  $0\% < \text{duty} < 100\%$ .

**Figure 18-34. Dead-Band Waveforms for Typical Cases ( $0\% < \text{Duty} < 100\%$ )**



The dead-band submodule supports independent values for rising-edge (RED) and falling-edge (FED) delays. The amount of delay is programmed using the DBRED and DBFED registers. These are 10-bit registers and their value represents the number of time-base clock, TBCLK, periods by which a signal edge is delayed. For example, the formula to calculate falling-edge-delay and rising-edge-delay is:

$$\begin{aligned} \text{FED} &= \text{DBFED} \times T_{\text{TBCLK}} \\ \text{RED} &= \text{DBRED} \times T_{\text{TBCLK}} \end{aligned}$$

Where  $T_{\text{TBCLK}}$  is the period of TBCLK, the prescaled version of EPWMCLK.

For convenience, delay values for various TBCLK options are shown in [Table 18-10](#). The ePWM input clock frequency that these delay values been computed by is 100 MHz.

**Table 18-10. Dead-Band Delay Values in  $\mu\text{S}$  as a Function of DBFED and DBRED**

Dead-Band Value DBFED, DBRED	Dead-Band Delay in $\mu\text{S}$		
	TBCLK = EPWMCLK/1	TBCLK = EPWMCLK /2	TBCLK = EPWMCLK/4
1	0.01 $\mu\text{S}$	0.02 $\mu\text{S}$	0.04 $\mu\text{S}$
5	0.05 $\mu\text{S}$	0.10 $\mu\text{S}$	0.20 $\mu\text{S}$
10	0.10 $\mu\text{S}$	0.20 $\mu\text{S}$	0.40 $\mu\text{S}$
100	1.00 $\mu\text{S}$	2.00 $\mu\text{S}$	4.00 $\mu\text{S}$
200	2.00 $\mu\text{S}$	4.00 $\mu\text{S}$	8.00 $\mu\text{S}$
400	4.00 $\mu\text{S}$	8.00 $\mu\text{S}$	16.00 $\mu\text{S}$
500	5.00 $\mu\text{S}$	10.00 $\mu\text{S}$	20.00 $\mu\text{S}$
600	6.00 $\mu\text{S}$	12.00 $\mu\text{S}$	24.00 $\mu\text{S}$
700	7.00 $\mu\text{S}$	14.00 $\mu\text{S}$	28.00 $\mu\text{S}$
800	8.00 $\mu\text{S}$	16.00 $\mu\text{S}$	32.00 $\mu\text{S}$
900	9.00 $\mu\text{S}$	18.00 $\mu\text{S}$	36.00 $\mu\text{S}$
1000	10.00 $\mu\text{S}$	20.00 $\mu\text{S}$	40.00 $\mu\text{S}$

When half-cycle clocking is enabled, the formula to calculate the falling-edge-delay and rising-edge-delay becomes:

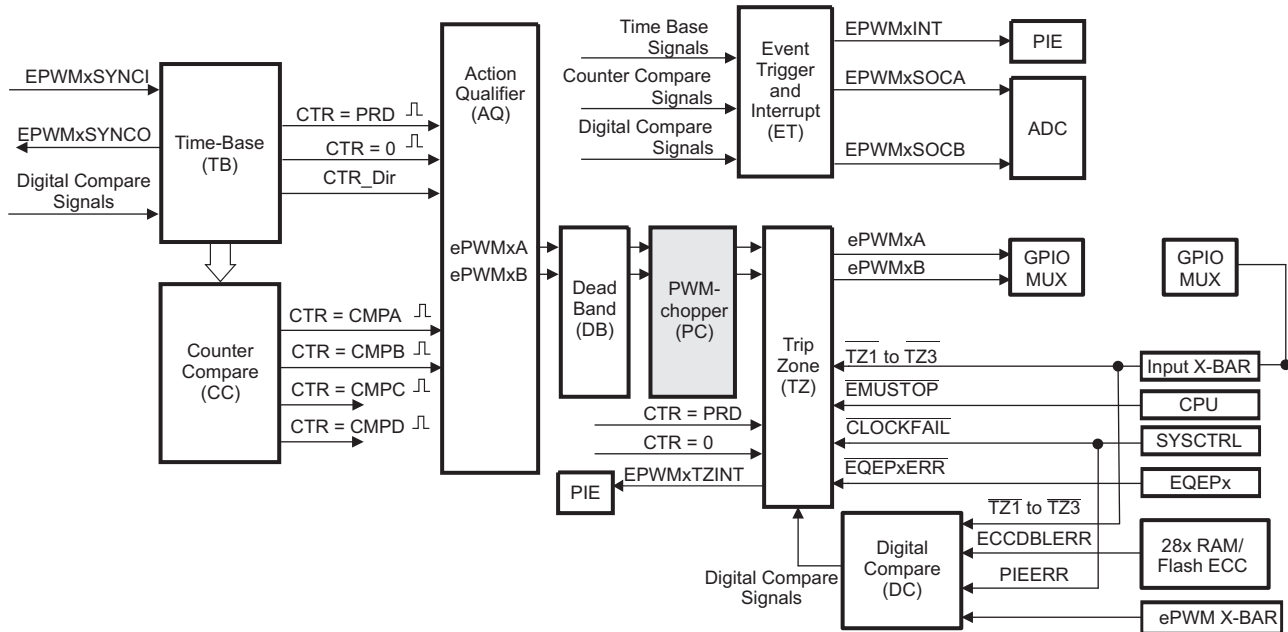
$$\begin{aligned} \text{FED} &= \text{DBFED} \times T_{\text{TBCLK}}/2 \\ \text{RED} &= \text{DBRED} \times T_{\text{TBCLK}}/2 \end{aligned}$$



## 18.8 PWM Chopper (PC) Submodule

Figure 18-35 illustrates the PWM chopper (PC) submodule within the ePWM module.

Figure 18-35. PWM Chopper Submodule



The PWM chopper submodule allows a high-frequency carrier signal to modulate the PWM waveform generated by the action-qualifier and dead-band submodules. This capability is important if you need pulse transformer-based gate drivers to control the power switching elements.

### 18.8.1 Purpose of the PWM Chopper Submodule

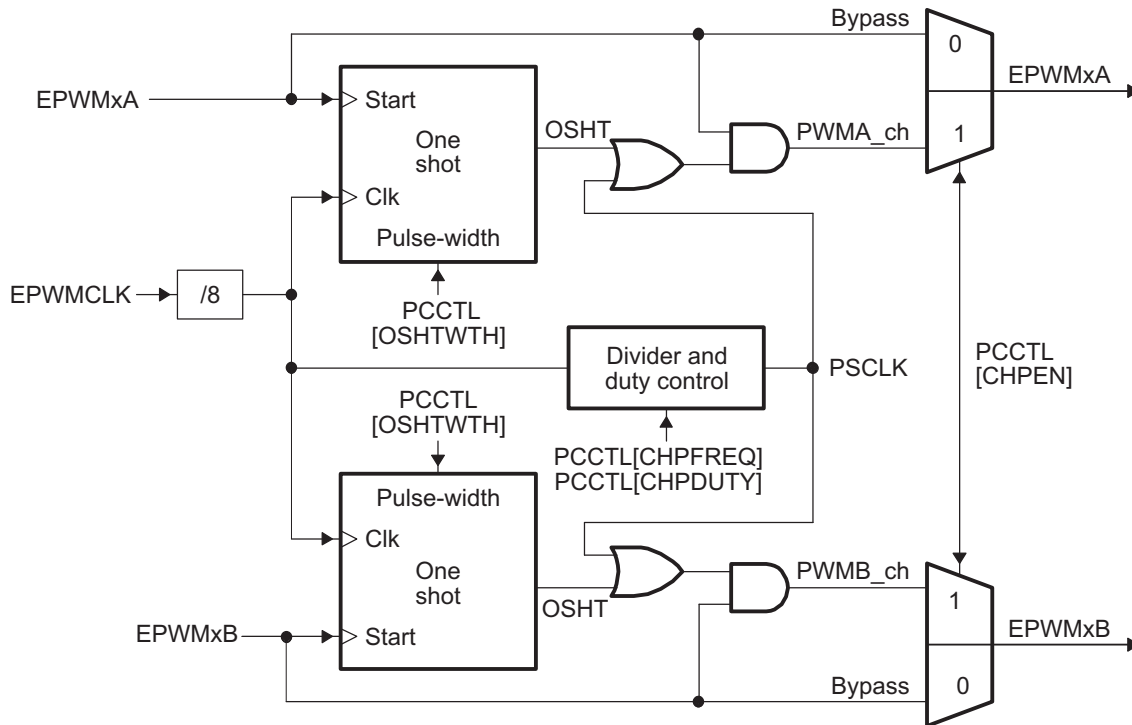
The key functions of the PWM chopper submodule are:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

### 18.8.2 Operational Highlights for the PWM Chopper Submodule

Figure 18-36 shows the operational details of the PWM chopper submodule. The carrier clock is derived from EPWMCLK. Its frequency and duty cycle are controlled via the CHPFREQ and CHPDUTY bits in the PCCTL register. The one-shot block is a feature that provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on. The one-shot width is programmed via the OSHTWTH bits. The PWM chopper submodule can be fully disabled (bypassed) via the CHPEN bit.

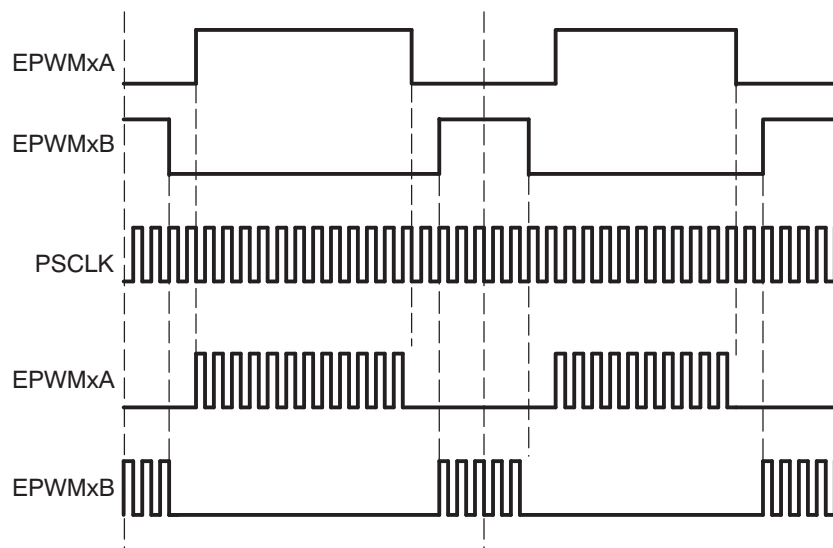
Figure 18-36. PWM Chopper Submodule Operational Details



### 18.8.3 Waveforms

Figure 18-37 shows simplified waveforms of the chopping action only; one-shot and duty-cycle control are not shown. Details of the one-shot and duty-cycle control are discussed in the following sections.

Figure 18-37. Simple PWM Chopper Submodule Waveforms Showing Chopping Action Only



### 18.8.3.1 One-Shot Pulse

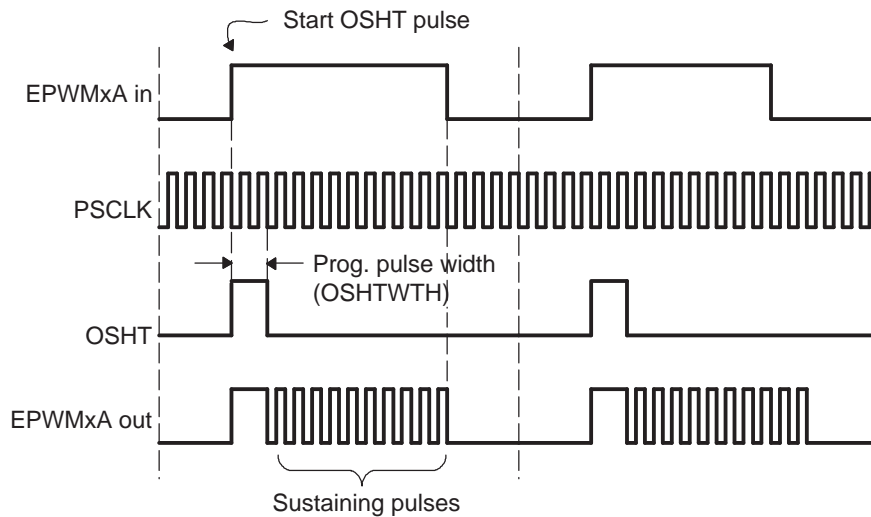
The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$$T_{1stpulse} = T_{EPWMCLK} \times 8 \times OSHTWTH$$

Where  $T_{EPWMCLK}$  is the period of the system clock (EPWMCLK) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 18-38 shows the first and subsequent sustaining pulses and Table 18-11 gives the possible pulse width values for a EPWMCLK = 80 MHz.

**Figure 18-38. PWM Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses**



**Table 18-11. Possible Pulse Width Values for EPWMCLK = 80 MHz**

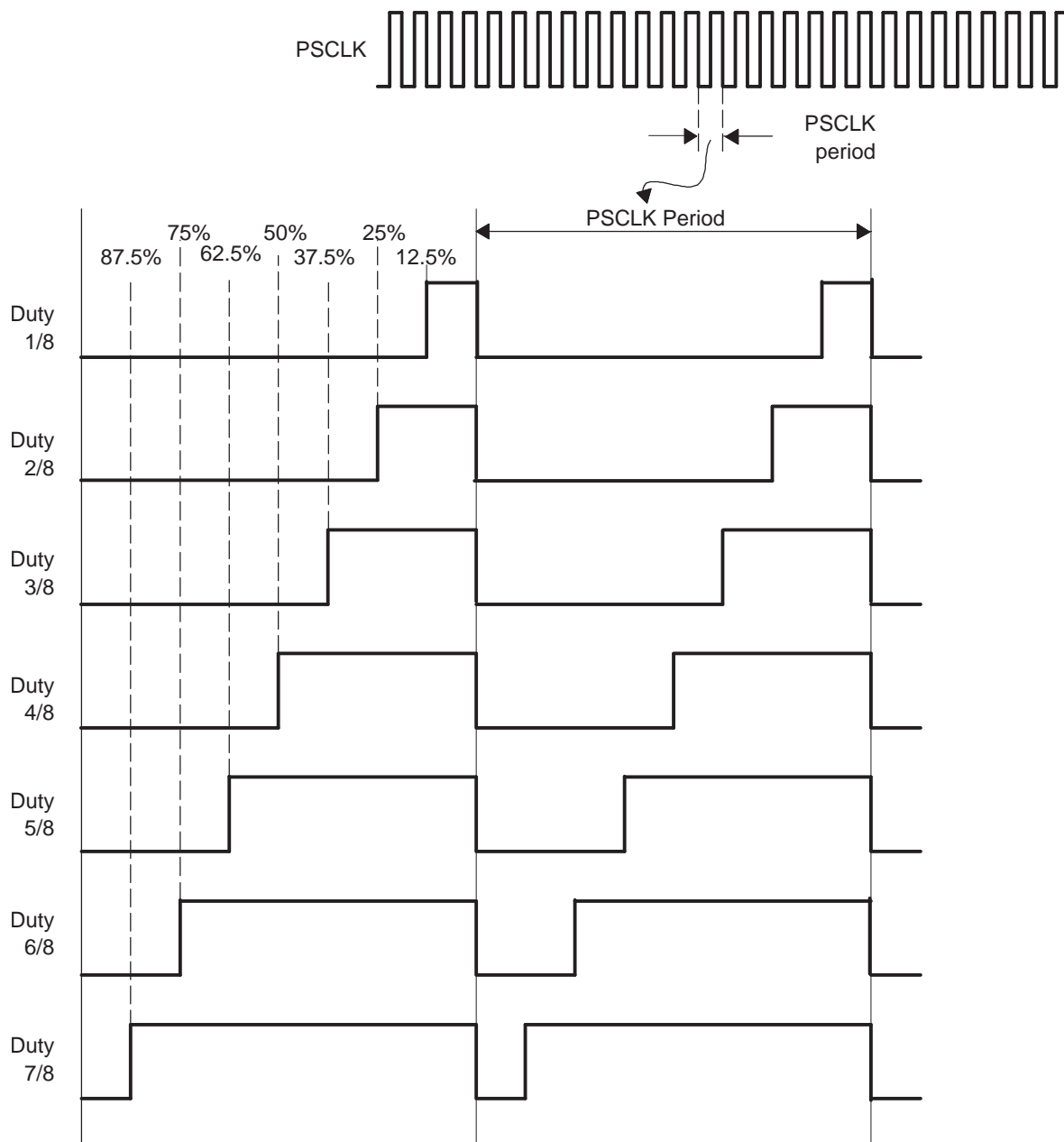
OSHTWTHz (hex)	Pulse Width (nS)
0	100
1	200
2	300
3	400
4	500
5	600
6	700
7	800
8	900
9	1000
A	1100
B	1200
C	1300
D	1400
E	1500
F	1600

### 18.8.3.2 Duty Cycle Control

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses ensure the correct drive strength and polarity is maintained on the power switch gate during the on period, and hence a programmable duty cycle allows a design to be tuned or optimized via software control.

Figure 18-39 shows the duty cycle control that is possible by programming the CHPDUTY bits. One of seven possible duty ratios can be selected ranging from 12.5% to 87.5%.

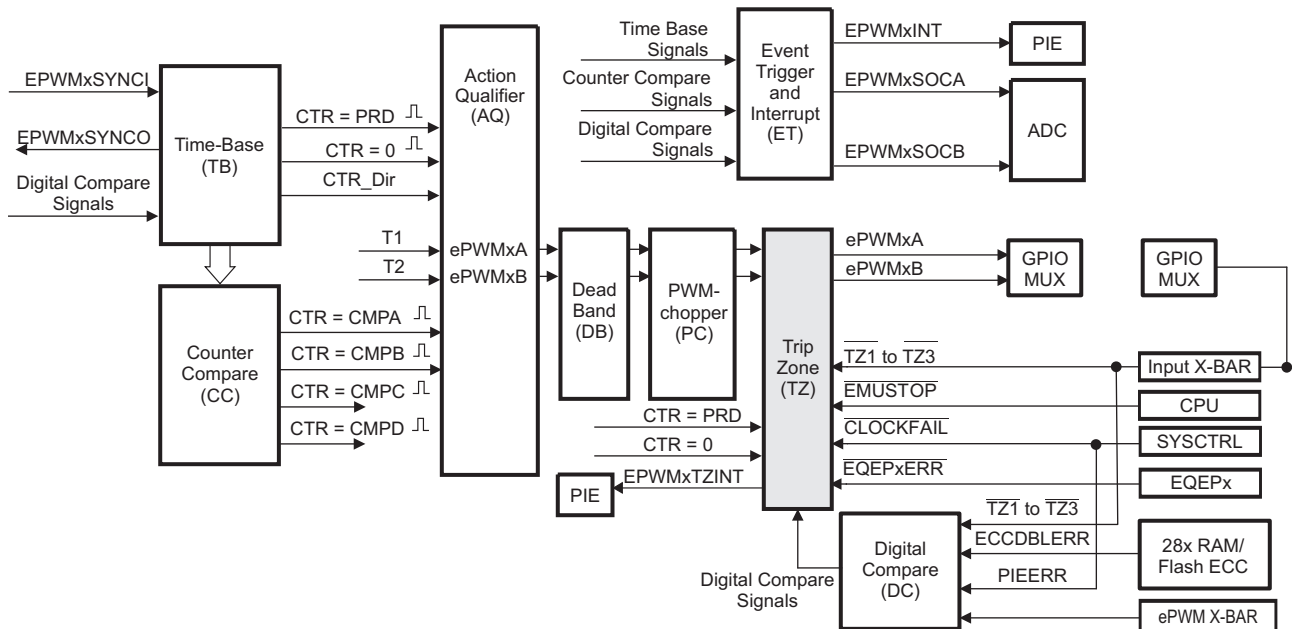
**Figure 18-39. PWM Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses**



## 18.9 Trip-Zone (TZ) Submodule

Figure 18-40 shows how the trip-zone (TZ) submodule fits within the ePWM module.

Figure 18-40. Trip-Zone Submodule



Each ePWM module is connected to six  $\overline{TZn}$  signals ( $\overline{TZ1}$  to  $\overline{TZ6}$ ).  $\overline{TZ1}$  to  $\overline{TZ3}$  are sourced from the GPIO mux.  $\overline{TZ4}$  is sourced from an inverted EQEPxERR signal on those devices with an EQEP module.  $\overline{TZ5}$  is connected to the system clock fail logic, and  $\overline{TZ6}$  is sourced from the EMUSTOP output from the CPU. These signals indicate external fault or trip conditions, and the ePWM outputs can be programmed to respond accordingly when faults occur.

### 18.9.1 Purpose of the Trip-Zone Submodule

The key functions of the trip-zone submodule are:

- Trip inputs  $\overline{TZ1}$  to  $\overline{TZ6}$  can be flexibly mapped to any ePWM module.
- Upon a fault condition, outputs EPWMxA and EPWMxB can be forced to one of the following:
  - High
  - Low
  - High-impedance
  - No action taken
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
- Support for cycle-by-cycle tripping (CBC) for current limiting operation.
- Support for digital compare tripping (DC) based on state of on-chip analog comparator module outputs and/or  $\overline{TZ1}$  to  $\overline{TZ3}$  signals.
- Each trip-zone input and digital compare (DC) submodule DCAEVT1/2 or DCBEVT1/2 force event can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone input.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

### 18.9.2 Operational Highlights for the Trip-Zone Submodule

The following sections describe the operational highlights and configuration options for the trip-zone submodule.

The trip-zone signals  $\overline{TZ1}$  to  $\overline{TZ6}$  (also collectively referred to as  $\overline{TZn}$ ) are active low input signals. When one of these signals goes low, or when a DCAEVT1/2 or DCBEVT1/2 force happens based on the TZDCSEL register event selection, it indicates that a trip event has occurred. Each ePWM module can be individually configured to ignore or use each of the trip-zone signals or DC events. Which trip-zone signals or DC events are used by a particular ePWM module is determined by the TZSEL register for that specific ePWM module. The trip-zone signals may or may not be synchronized to the ePWMclock (EPWMCLK) and digitally filtered within the GPIO MUX block. A minimum of  $3 \cdot TBCLK$  low pulse width on  $\overline{TZn}$  inputs is sufficient to trigger a fault condition on the ePWM module. If the pulse width is less than this, the trip condition may not be latched by CBC or OST latches. The asynchronous trip makes sure that if clocks are missing for any reason, the outputs can still be tripped by a valid event present on  $\overline{TZn}$  inputs. The GPIOs or peripherals must be appropriately configured. For more information, see the device-specific version of the *System Control and Interrupts* chapter.

Each  $\overline{TZn}$  input can be individually configured to provide either a cycle-by-cycle or one-shot trip event for an ePWM module. DCAEVT1 and DCBEVT1 events can be configured to directly trip an ePWM module or provide a one-shot trip event to the module. Likewise, DCAEVT2 and DCBEVT2 events can also be configured to directly trip an ePWM module or provide a cycle-by-cycle trip event to the module. This configuration is determined by the TZSEL[DCAEVT1/2], TZSEL[DCBEVT1/2], TZSEL[CBCn], and TZSEL[OSHTn] control bits (where n corresponds to the trip input) respectively.

- **Cycle-by-Cycle (CBC):**

When a cycle-by-cycle trip event occurs, the action specified in the TZCTL[TZA] and TZCTL[TZB] bits is carried out immediately on the EPWMxA and/or EPWMxB outputs. [Table 18-12](#) lists some of the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up and/or while it is counting down by appropriately configuring bits in the TZCTL2 register. Actions specified in the TZCTL2 register take effect only when the ETZE bit in TZCTL2 is set.

Additionally, when a cycle-by-cycle trip event occurs, the cycle-by-cycle trip event flag (TZFLG[CBC]) is set and a EPWMx\_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral. A corresponding flag for the event that caused the CBC event is also set in register TZCBCFLG.

If the CBC interrupt is enabled via the TZEINT register, and DCAEVT2 or DCBEVT2 are selected as CBC trip sources via the TZSEL register, it is not necessary to also enable the DCAEVT2 or DCBEVT2 interrupts in the TZEINT register, as the DC events trigger interrupts through the CBC mechanism.

The specified condition on the inputs is automatically cleared based on the selection made with TZCLR[CBCPULSE] if the trip event is no longer present. Therefore, in this mode, the trip event is cleared or reset every PWM cycle. The TZFLG[CBC] and TZCBCFLG flag bits will remain set until they are manually cleared by writing to the TZCLR[CBC] and TZCBCCLR flag bits. If the cycle-by-cycle trip event is still present when the TZFLG[CBC] and/or TZCBCFLG register bits are cleared, then these bits will again be immediately set..

- **One-Shot (OSHT):**

When a one-shot trip event occurs, the action specified in the TZCTL[TZA] and TZCTL[TZB] bits is carried out immediately on the EPWMxA and/or EPWMxB output. [Table 18-12](#) lists some of the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up and/or while it is counting down by appropriately configuring bits in TZCTL2 register. Actions specified in TZCTL2 register take effect only when ETZE bit in TZCTL2 is set.

Additionally, when a one-shot trip event occurs, the one-shot trip event flag (TZFLG[OST]) is set and a EPWMx\_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral. A corresponding flag for the event that caused the OST event is also set in register TZOSTFLG. The one-shot trip condition must be cleared manually by writing to the TZCLR[OST] bit. If desired, TZOSTFLG register bit should be cleared by manually writing to the corresponding bit in the TZOSTCLR register.

If the one-shot interrupt is enabled via the TZEINT register, and DCAEVT1 or DCBEVT1 are selected as OSHT trip sources via the TZSEL register, it is not necessary to also enable the DCAEVT1 or DCBEVT1 interrupts in the TZEINT register, as the DC events trigger interrupts through the OSHT mechanism.

- **Digital Compare Events (DCAEVT1/2 and DCBEVT1/2):**

A digital compare DCAEVT1/2 or DCBEVT1/2 event is generated based on a combination of the

DCAH/DCAL and DCBH/DCBL signals as selected by the TZDCSEL register. The signals which source the DCAH/DCAL and DCBH/DCBL signals are selected via the DCTRIPSEL register and can be either trip zone input pins or analog comparator CMPSSx signals. For more information on the digital compare submodule signals, see [Section 18.11](#).

When a digital compare event occurs, the action specified in the TZCTL[DCAEVT1/2] and TZCTL[DCBEVT1/2] bits is carried out immediately on the EPWMxA and/or EPWMxB output. [Table 18-12](#) lists the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up and/or while it is counting down by appropriately configuring bits in TZCTLDCA and TZCTLDCB register. Actions specified in TZCTLDCA and TZCTLDCB registers take effect only when ETZE bit in TZCTL2 is set.

In addition, the relevant DC trip event flag (TZFLG[DCAEVT1/2] / TZFLG[DCBEVT1/2]) is set and a EPWMx\_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral.

The specified condition on the pins is automatically cleared when the DC trip event is no longer present. The TZFLG[DCAEVT1/2] or TZFLG[DCBEVT1/2] flag bit will remain set until it is manually cleared by writing to the TZCLR[DCAEVT1/2] or TZCLR[DCBEVT1/2] bit. If the DC trip event is still present when the TZFLG[DCAEVT1/2] or TZFLG[DCBEVT1/2] flag is cleared, then it will again be immediately set.

The action taken when a trip event occurs can be configured individually for each of the ePWM output pins by way of the TZCTL, TZCTL2, TZCTLDCA, and TZCTLDCB register bit fields. Some of the possible actions, shown in the table below, can be taken on a trip event.

**Table 18-12. Possible Actions On a Trip Event**

TZCTL Register bit-field Settings	EPWMxA and/or EPWMxB	Comment
0,0	High-Impedance	Tripped
0,1	Force to High State	Tripped
1,0	Force to Low State	Tripped
1,1	No Change	Do Nothing. No change is made to the output.

### Example 18-1. Trip-Zone Configurations

#### Scenario A:

A one-shot trip event on  $\overline{TZ1}$  pulls both EPWM1A, EPWM1B low and also forces EPWM2A and EPWM2B high.

- Configure the ePWM1 registers as follows:
  - TZSEL[OSHT1] = 1: enables  $\overline{TZ1}$  as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables  $\overline{TZ1}$  as a one-shot event source for ePWM2
  - TZCTL[TZA] = 1: EPWM2A will be forced high on a trip event.
  - TZCTL[TZB] = 1: EPWM2B will be forced high on a trip event.

#### Scenario B:

A cycle-by-cycle event on  $\overline{TZ5}$  pulls both EPWM1A, EPWM1B low.  
A one-shot event on  $\overline{TZ1}$  or  $\overline{TZ6}$  puts EPWM2A into a high impedance state.

- Configure the ePWM1 registers as follows:
  - TZSEL[CBC5] = 1: enables  $\overline{TZ5}$  as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.

**Example 18-1. Trip-Zone Configurations (continued)**

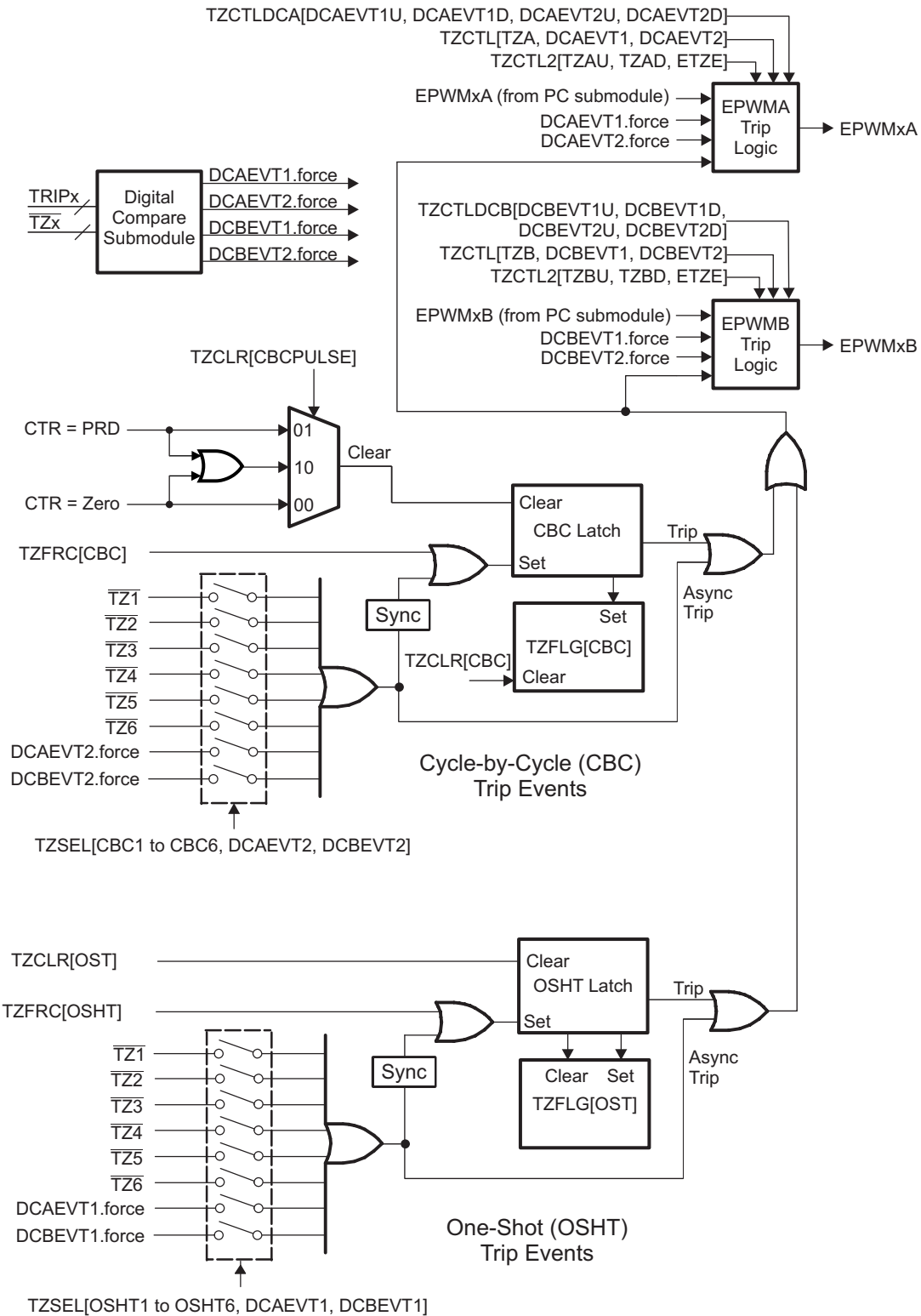
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables  $\overline{TZ1}$  as a one-shot event source for ePWM2
  - TZSEL[OSHT6] = 1: enables  $\overline{TZ6}$  as a one-shot event source for ePWM2
  - TZCTL[TZA] = 0: EPWM2A will be put into a high-impedance state on a trip event.
  - TZCTL[TZB] = 3: EPWM2B will ignore the trip event.

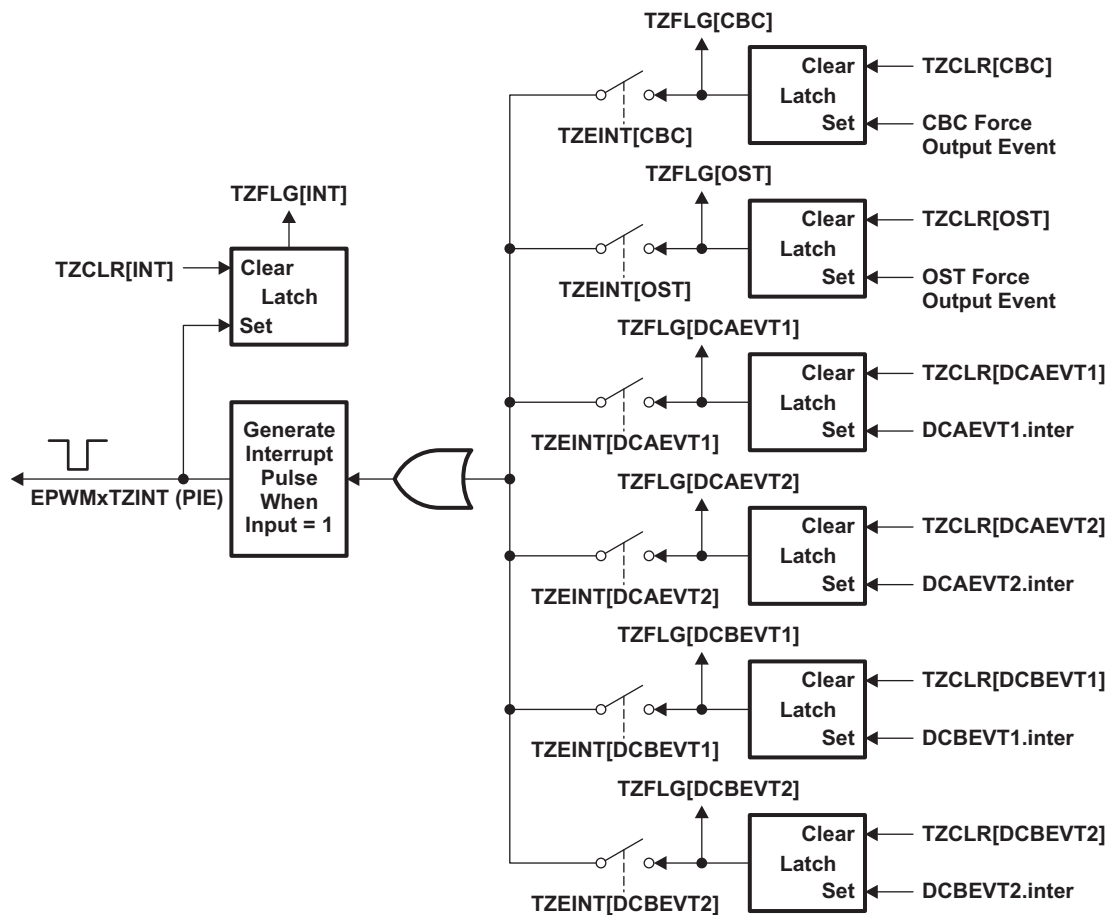
**18.9.3 Generating Trip Event Interrupts**

[Figure 18-41](#) and [Figure 18-42](#) illustrate the trip-zone submodule control and interrupt logic, respectively. DCAEVT1/2 and DCBEVT1/2 signals are described in further detail in [Section 18.11](#).



Figure 18-41. Trip-Zone Submodule Mode Control Logic



**Figure 18-42. Trip-Zone Submodule Interrupt Logic**


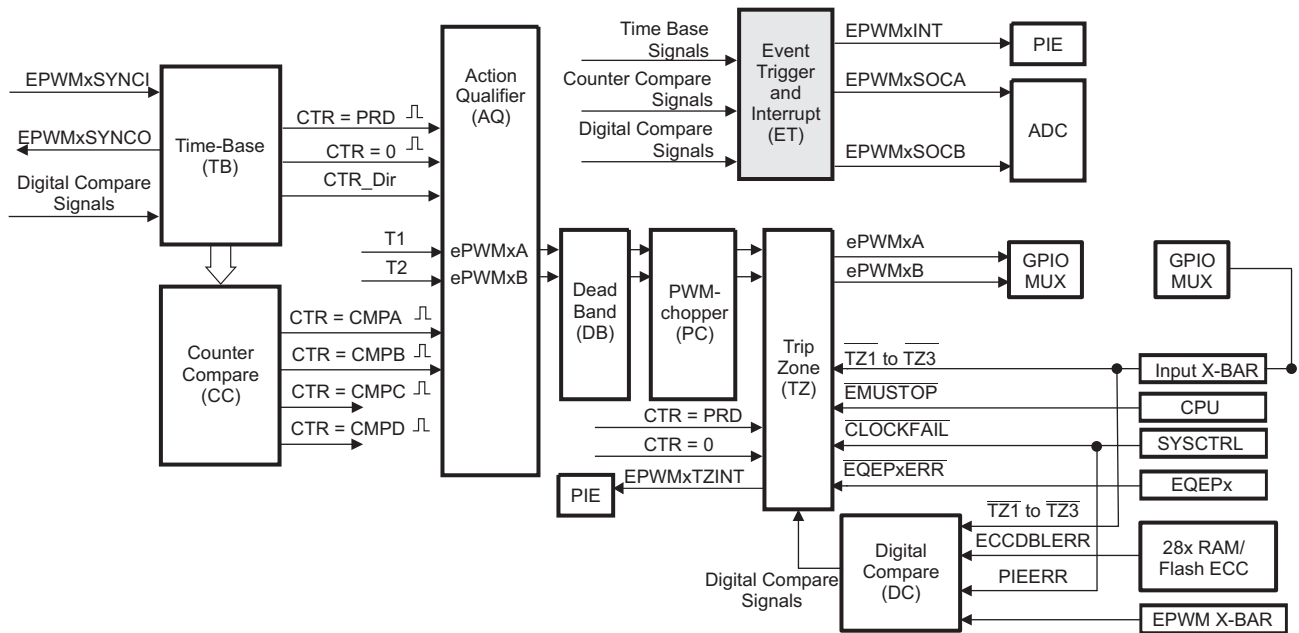
## 18.10 Event-Trigger (ET) Submodule

The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base, counter-compare, and digital-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests and ADC start of conversion at:
  - Every event
  - Every second event
  - Up to every fifteenth event
- Provides full visibility of event generation via event counters and flags
- Allows software forcing of Interrupts and ADC start of conversion

The event-trigger submodule manages the events generated by the time-base submodule, the counter-compare submodule, and the digital-compare submodule to generate an interrupt to the CPU and/or a start of conversion pulse to the ADC when a selected event occurs. [Figure 18-43](#) illustrates where the event-trigger submodule fits within the ePWM system.

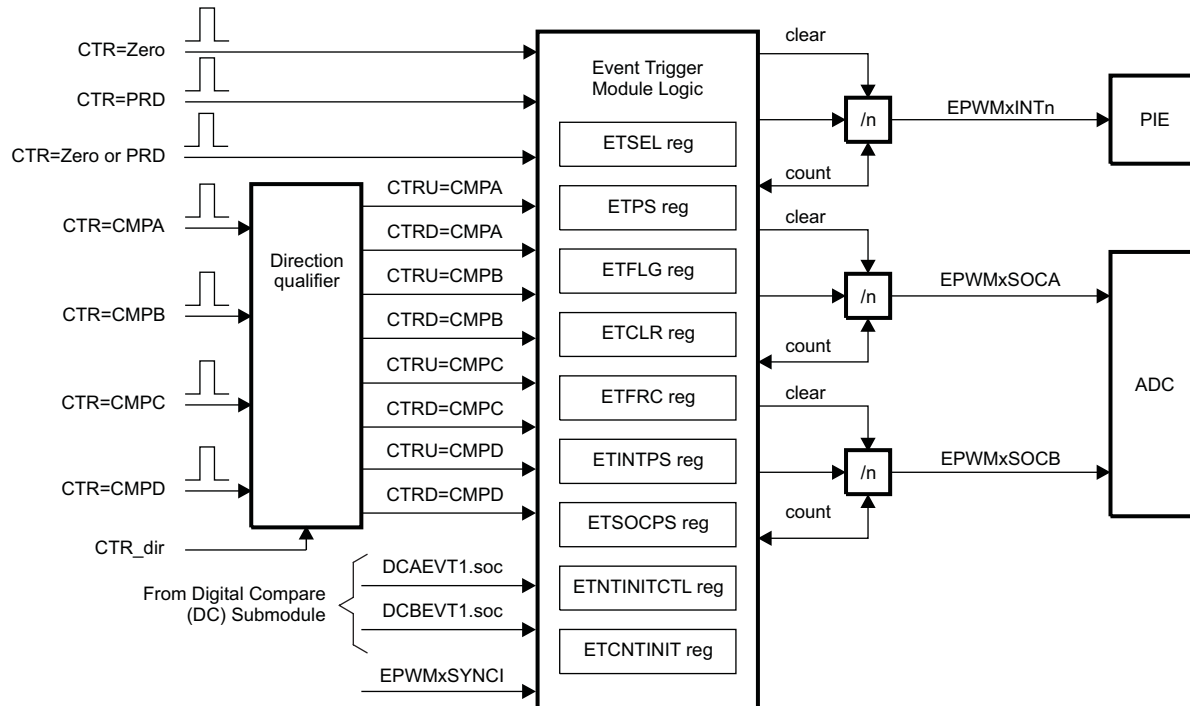
Figure 18-43. Event-Trigger Submodule



### 18.10.1 Operational Overview of the ePWM Type 4 Event-Trigger Submodule

The event-trigger submodule monitors various event conditions (shown as inputs on the left side of Figure 18-44) and can be configured to prescale these events before issuing an Interrupt request or an ADC start of conversion. The event-trigger prescaling logic can issue Interrupt requests and ADC start of conversion at:

- Every event
- Every second event
- Up to Every fifteenth event

**Figure 18-44. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs**


- ETSEL - This selects which of the possible events will trigger an interrupt or start an ADC conversion.
- ETPS - This programs the event prescaling options mentioned above.
- ETFLG - These are flag bits indicating status of the selected and prescaled events.
- ETCLR - These bits allow you to clear the flag bits in the ETFLG register via software.
- ETFRC - These bits allow software forcing of an event. Useful for debugging or software intervention.
- ETINTPS - This programs the interrupt event prescaling options, supporting count and period up to 15 events.
- ETSOCPS - This programs the SOC event prescaling options, supporting count and period up to 15 events.
- ETCNTINITCTL - These bits enable ETCNTINIT initialization via SYNC event OR via software force.
- ETCNTINIT - These bits allow you to initialize INT/SOCA/SOCB counters on SYNC events (or software force) with user programmed value.

A more detailed look at how the various register bits interact with the Interrupt and ADC start of conversion logic are shown in [Figure 18-45](#), [Figure 18-46](#), and [Figure 18-47](#).

[Figure 18-45](#) shows the event-trigger's interrupt generation logic. The interrupt-period (ETPS[INTPRD]) bits specify the number of events required to cause an interrupt pulse to be generated. The choices available are:

- Do not generate an interrupt.
- Generate an interrupt on every event
- Generate an interrupt on every second event
- Generate an interrupt on every third event

On ePWM type 4, in order to enable event generation capability up to 15 events the following changes have been made. The selection made on ETPS[INTPSSSEL] bit determines whether ETINTPS register, INTCNT2 and INTPRD2 bit fields determine frequency of events (interrupt once every 0-15 events).

Which event can cause an interrupt is configured by the interrupt selection (ETSEL[INTSEL]) and (ETSEL[INTSELCMP]) bits. The event can be one of the following:

- Time-base counter equal to zero (TBCTR = 0x00).
- Time-base counter equal to period (TBCTR = TBPRD).
- Time-base counter equal to zero or period (TBCTR = 0x00 || TBCTR = TBPRD).
- Time-base counter equal to the compare A register (CMPA) when the timer is incrementing.
- Time-base counter equal to the compare A register (CMPA) when the timer is decrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is incrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is decrementing.
- Time-base counter equal to the compare C register (CMPC) when the timer is incrementing.
- Time-base counter equal to the compare C register (CMPC) when the timer is decrementing.
- Time-base counter equal to the compare D register (CMPD) when the timer is incrementing.
- Time-base counter equal to the compare D register (CMPD) when the timer is decrementing.

The number of events that have occurred can be read from the interrupt event counter ETPS[INTCNT] or ETINTPS[INTCNT2] register bits based off of the selection made using ETPS[INTPSSEL]. That is, when the specified event occurs the ETPS[INTCNT] or ETINTPS[INTCNT2] bits are incremented until they reach the value specified by ETPS[INTPRD] or ETINTPS[INTPRD2] determined again by the selection made in ETPS[INTPSSEL]. When ETPS[INTCNT] = ETPS[INTPRD] the counter stops counting and its output is set. The counter is only cleared when an interrupt is sent to the PIE.

When ETPS[INTCNT] reaches ETPS[INTPRD] the following behavior will occur [The below behavior is also applicable to ETINTPS[INTCNT2] & ETINTPS[INTPRD2] :

- If interrupts are enabled, ETSEL[INTEN] = 1 and the interrupt flag is clear, ETFLG[INT] = 0, then an interrupt pulse is generated and the interrupt flag is set, ETFLG[INT] = 1, and the event counter is cleared ETPS[INTCNT] = 0. The counter will begin counting events again.
- If interrupts are disabled, ETSEL[INTEN] = 0, or the interrupt flag is set, ETFLG[INT] = 1, the counter stops counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
- If interrupts are enabled, but the interrupt flag is already set, then the counter will hold its output high until the ENTFLG[INT] flag is cleared. This allows for one interrupt to be pending while one is serviced.

Writing to the INTPRD bits will automatically clear the counter INTCNT = 0 and the counter output will be reset (so no interrupts are generated). Writing a 1 to the ETFRC[INT] bit will increment the event counter INTCNT. The counter will behave as described above when INTCNT = INTPRD. When INTPRD = 0, the counter is disabled and hence no events will be detected and the ETFRC[INT] bit is also ignored. The same applies to ETINTPS[INTCNT2] & ETINTPS[INTPRD2]

The above definition means that you can generate an interrupt on every event, on every second event, or on every third event if using the INTCNT and INTPRD. You can generate an interrupt on every event up to 15 events if using the INTCNT2 and INTPRD2.

The INTCNT2 value can be initialized with the value from ETCNTINIT[INTINIT] based on the selection made in ETCNTINITCTL[INTINITEN]. When ETCNTINITCTL[INTINITEN] is set, then it enables initialization of INTCNT2 counter with contents of ETCNTINIT[INTINIT] on a SYNC event or software force determined by ETCNTINITCTL[INTINITFRC] .

Figure 18-45. Event-Trigger Interrupt Generator

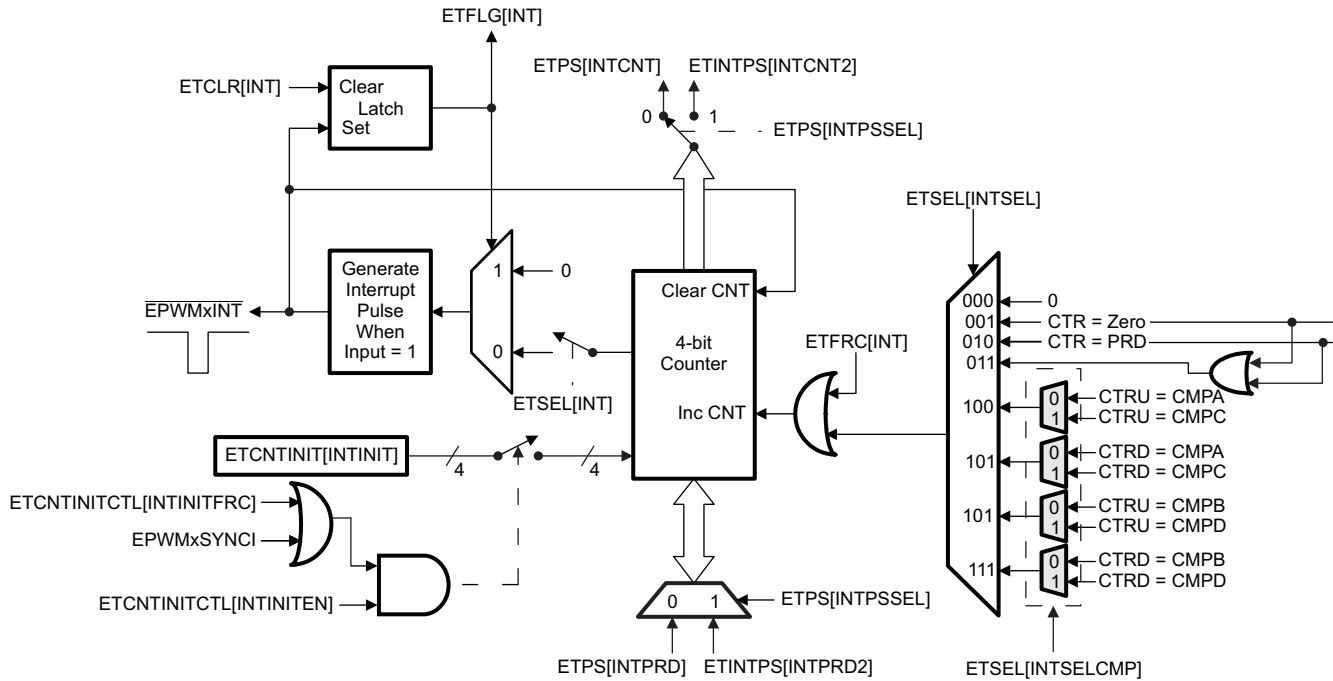
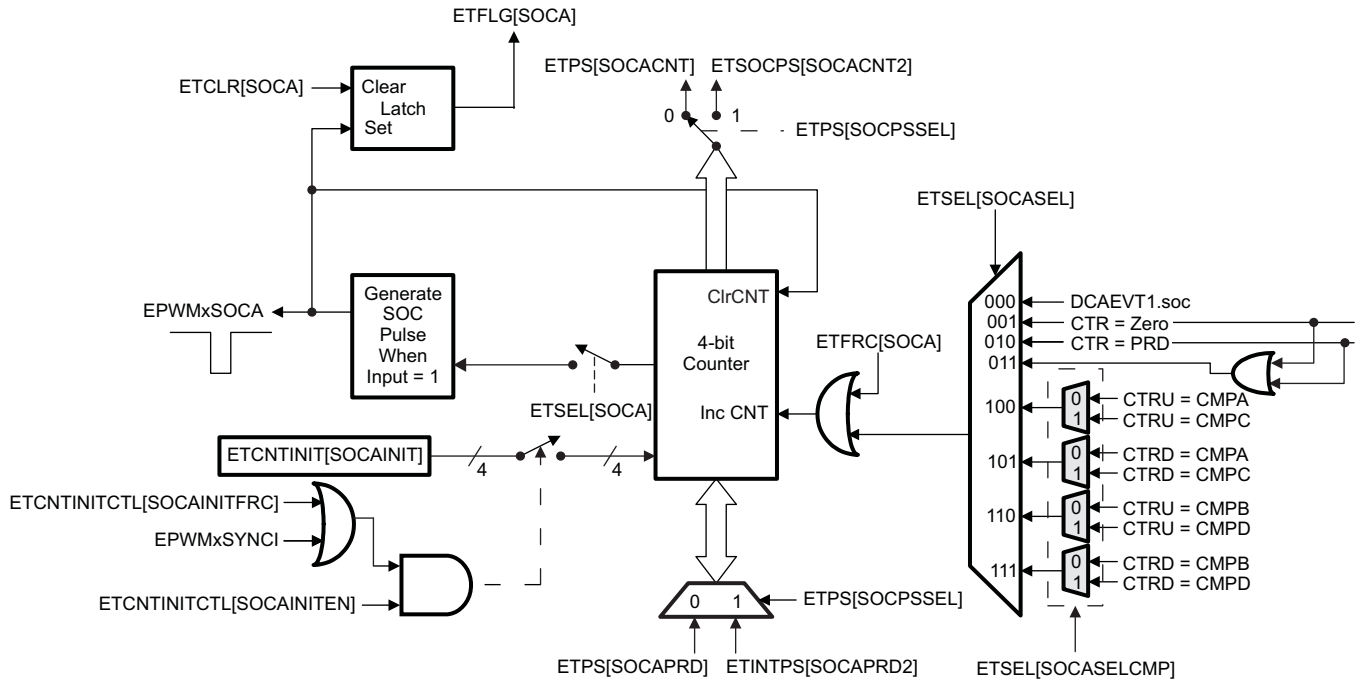


Figure 18-46 shows the operation of the event-trigger's start-of-conversion-A (SOCA) pulse generator. The enhancements include SOCASELCMP and SOCBSELCMP bit fields defined in the ETSEL register enable CMPC and CMPD events respectively to cause a start of conversion. The ETPS[SOCPSSEL] bit field determines whether SOCACNT2 and SOCAPRD2 take control or not. The ETPS[SOCACNT] counter and ETPS[SOCAPRD] period values behave similarly to the interrupt generator except that the pulses are continuously generated. That is, the pulse flag ETFLG[SOCA] is latched when a pulse is generated, but it does not stop further pulse generation. The enable/disable bit ETSEL[SOCAEN] stops pulse generation, but input events can still be counted until the period value is reached as with the interrupt generation logic. The event that will trigger an SOCA and SOCB pulse can be configured separately in the ETSEL[SOCASEL] and ETSEL[SOCBSEL] bits. The possible events are the same events that can be specified for the interrupt generation logic with the addition of the DCAEVT1.soc and DCBEVT1.soc event signals from the digital compare (DC) submodule. The SOCACNT2 initialization scheme is very similar to the interrupt generator with respective enable, value initialize and SYNC or software force options.

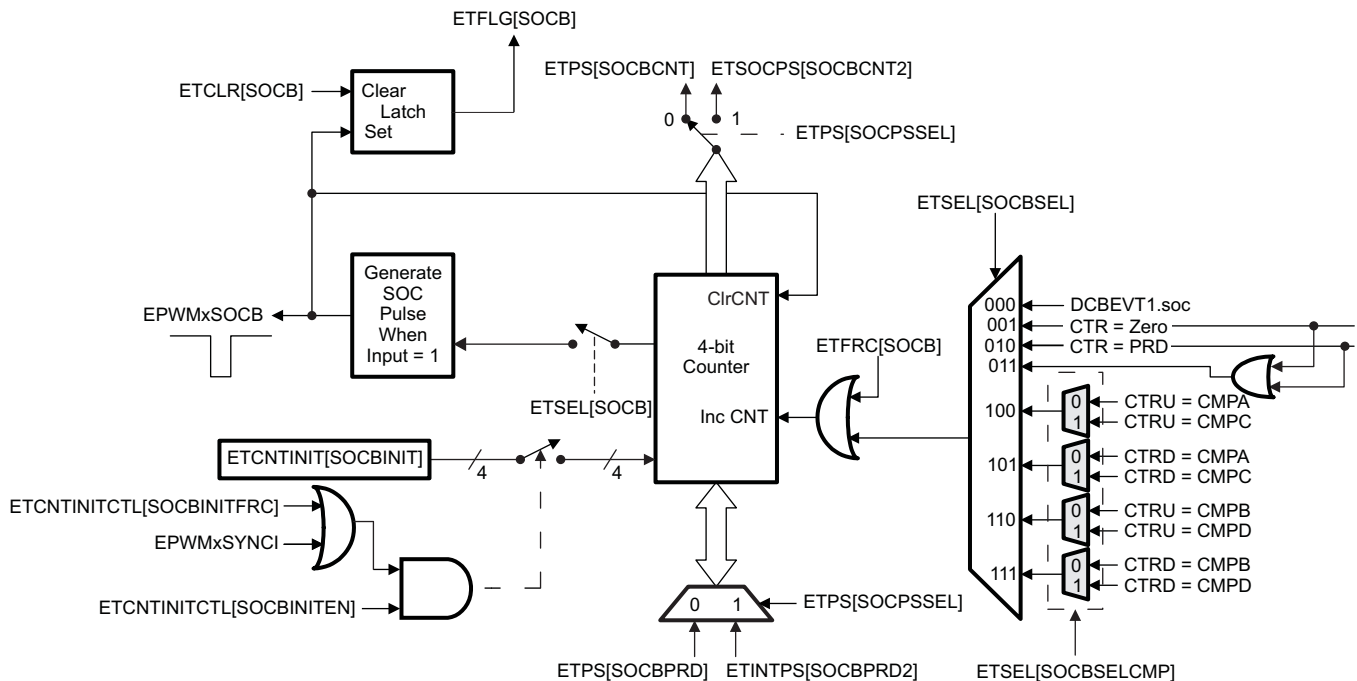
Figure 18-46. Event-Trigger SOCA Pulse Generator



A The DCAEVT1.soc signals are signals generated by the Digital compare (DC) submodule in Section 18.11.

Figure 18-47 shows the operation of the event-trigger's start-of-conversion-B (SOCB) pulse generator. The event-trigger's SOCB pulse generator operates the same way as the SOCA.

Figure 18-47. Event-Trigger SOCB Pulse Generator

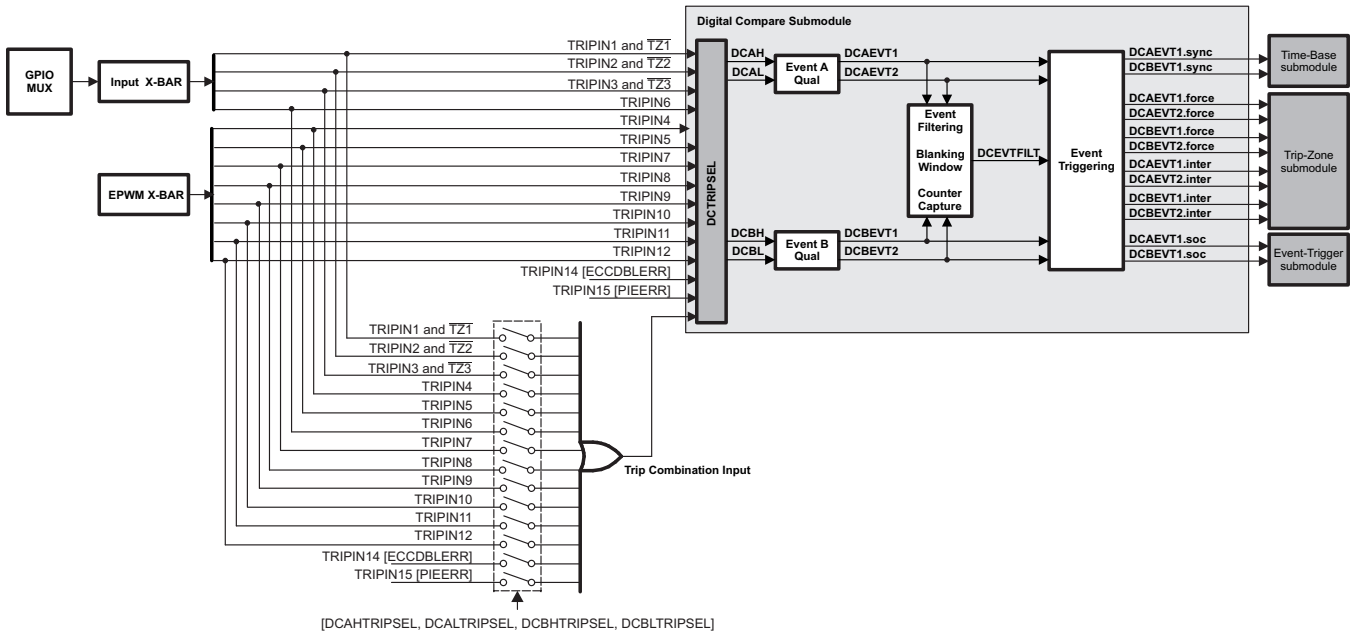


A The DCBEVT1.soc signals are signals generated by the Digital compare (DC) submodule in Section 18.11.

### 18.11 Digital Compare (DC) Submodule

Figure 18-48 illustrates where the digital compare (DC) submodule signals interface to other submodules in the ePWM system.

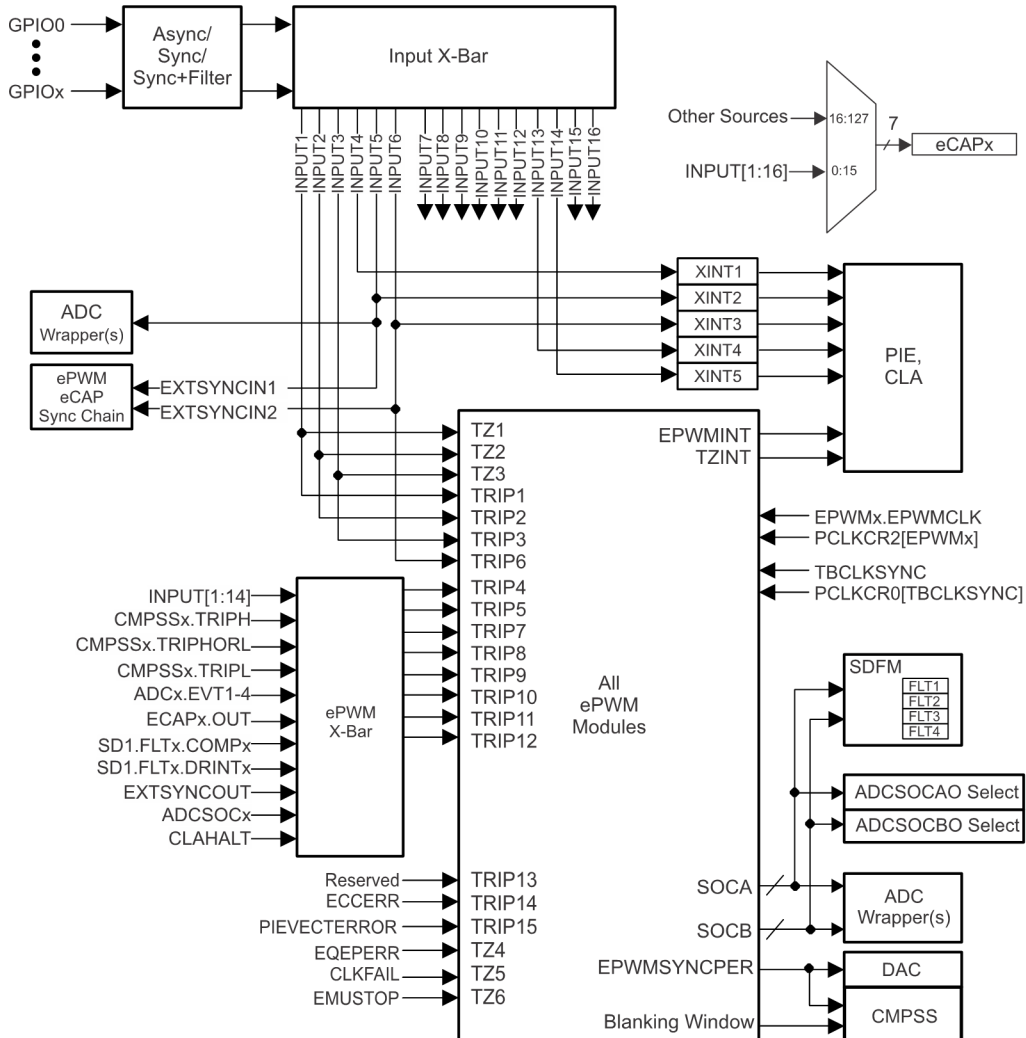
Figure 18-48. Digital-Compare Submodule High-Level Block Diagram



The eCAP input signals are sourced from the Input X-BAR signals as shown in Figure 18-49.



Figure 18-49. GPIO MUX-to-Trip Input Connectivity



On this device, any of the GPIO pins can be flexibly mapped to be the trip-zone input and/or trip inputs to the trip-zone submodule and digital compare submodule. The Input X-BAR Input Select (INPUTxSELECT) register defines which GPIO pins gets assigned to be the trip-zone inputs / trip inputs.

The digital compare (DC) submodule compares signals external to the ePWM module (for instance, CMPSSx signals from the analog comparators) to directly generate PWM events/actions which then feed to the event-trigger, trip-zone, and time-base submodules. Additionally, blanking window functionality is supported to filter noise or unwanted pulses from the DC event signals.

**NOTE:** The user is responsible for driving correct state on the selected pin before enabling clock and configuring the trip input for the respective ePWM peripheral to avoid spurious latch of TRIP signal.

### 18.11.1 Purpose of the Digital Compare Submodule

The key functions of the digital compare submodule are:

- Analog comparator (COMP) module outputs fed through the Input X-BAR logic externally using the GPIO peripheral, internal PIE, ECC error signals, TZ1, TZ2, and TZ3 inputs generate Digital Compare A High/Low (DCAH, DCAL) and Digital Compare B High/Low (DCBH, DCBL) signals.
- DCAH/L and DCBH/L signals trigger events which can then either be filtered or fed directly to the trip-

zone, event-trigger, and time-base submodules to:

- generate a trip zone interrupt
- generate an ADC start of conversion
- force an event
- generate a synchronization event for synchronizing the ePWM module TBCTR.
- Event filtering (blanking window logic) can optionally blank the input signal to remove noise.

### 18.11.2 Enhanced Trip Action Using CMPSS

In order to allow multiple CMPSS at a time to affect DCA/BEVTx events and trip actions, there is a OR logic to bring together ALL trip inputs (up to 15) from sources external to the ePWM module and feed into DCAH, DCAL, DCBH, and DCBL as a “combinational input” using the DCTRIPSEL register. This is configured by selecting “Trip combination input” (value of 0xF) in the DCTRIPSEL register.

The user has a discrete choice of which trip inputs to put through the combinational logic for generating the DCAH, DCAL, DCBH and DCBL signals. This is achieved using the DCAHTRIPSEL, DCALTRIPSEL, DCBHTRIPSEL and DCBLTRIPSEL register selections. Inputs selected for combinational input are passed through to the DCTRIPSEL register.

### 18.11.3 Using CMPSS to Trip the ePWM on a Cycle-by-Cycle Basis

When using the CMPSS to trip the ePWM on a cycle-by-cycle basis, steps should be taken to prevent an asserted comparator trip state in one PWM cycle from extending into the following cycle. The CMPSS can be used to signal a trip condition to the downstream ePWM modules. For applications like peak current mode control, only one trip event per PWM cycle is expected. Under certain conditions, it is possible for a sustained or late trip event (arriving near the end of a PWM cycle) to carry over into the next PWM cycle if precautions are not taken. If either the CMPSS Digital Filter or the ePWM Digital Compare (DC) submodule is configured to qualify the comparator trip signal, “N” number of clock cycles of qualification will be introduced before the ePWM trip logic can respond to logic changes of the trip signal. Once an ePWM trip condition is qualified, the trip condition will remain active for N clock cycles after the comparator trip signal has de-asserted. If a qualified comparator trip signal remains asserted within N clock cycles prior to the end of a PWM cycle, the trip condition will not be cleared until after the following PWM cycle has started. Thus, the new PWM cycle will detect a trip condition as soon as it begins.

To avoid this undesired trip condition, the user application should take steps to ensure that the qualified trip signal seen by the ePWM trip logic is deasserted prior to the end of each PWM cycle. This can be accomplished through various methods:

- Design the system such that a comparator trip will not be asserted within N clock cycles prior to the end of the PWM cycle.
- Activate blanking of the comparator trip signal via the ePWM event filter at least two clock cycles prior to the PWMSYNCPER signal and continue blanking for at least N clock cycles into the next PWM cycle.
- If the CMPSS COMPxLATCH path is used, clear the COMPxLATCH at least N clock cycles prior to the end of the PWM cycle. The latch can be cleared by software (via COMPSTSCLR) or by generating an early PWMSYNCPER signal. The ePWM modules on this device include the ability to generate PWMSYNCPER upon a CMPC or CMPD match (via HRPCTL) for arbitrary PWMSYNCPER placement within the PWM cycle.

### 18.11.4 Operation Highlights of the Digital Compare Submodule

The following sections describe the operational highlights and configuration options for the digital compare submodule.

### 18.11.4.1 Digital Compare Events

As illustrated in [Section 18.11.4.2](#) earlier in this section, trip zone inputs ( $\overline{TZ1}$ ,  $\overline{TZ2}$ , and  $\overline{TZ3}$ ) and CMPSSx signals from the analog comparator (COMP) module can be selected via the DCTRIPSEL bits to generate the Digital Compare A High and Low (DCAH/L) and Digital Compare B High and Low (DCBH/L) signals. Then, the configuration of the TZDSEL register qualifies the actions on the selected DCAH/L and DCBH/L signals, which generate the DCAEVT1/2 and DCBEVT1/2 events (Event Qualification A and B).

---

**NOTE:** The  $\overline{TZn}$  signals, when used as a DCEVT tripping functions, are treated as a normal input signal and can be defined to be active high or active low inputs. ePWM outputs are asynchronously tripped when either the  $\overline{TZn}$ , DCAEVTx.force, or DCBEVTx.force signals are active. For the condition to remain latched, a minimum of  $3 \cdot TBCLK$  sync pulse width is required. If pulse width is  $< 3 \cdot TBCLK$  sync pulse width, the trip condition may or may not get latched by CBC or OST latches.

---

The DCAEVT1/2 and DCBEVT1/2 events can then be filtered to provide a filtered version of the event signals (DCEVTFILT) or the filtering can be bypassed. Filtering is discussed further in [Section 18.11.4.2](#). Either the DCAEVT1/2 and DCBEVT1/2 event signals or the filtered DCEVTFILT event signals can generate a force to the trip zone module, a TZ interrupt, an ADC SOC, or a PWM sync signal.

- **force signal:**

DCAEVT1/2.force signals force trip zone conditions which either directly influence the output on the EPWMxA pin (via TZCTL, TZCTLDCA, TZCTLDCB register configurations) or, if the DCAEVT1/2 signals are selected as one-shot or cycle-by-cycle trip sources (via the TZSEL register), the DCAEVT1/2.force signals can effect the trip action via the TZCTL or TZCTL2 register configurations. The DCBEVT1/2.force signals behaves similarly, but affect the EPWMxB output pin instead of the EPWMxA output pin.

The priority of conflicting actions on the TZCTL, TZCTL2, TZCTLDCA and TZCTLDCB registers is as follows (highest priority overrides lower priority):

Output EPWMxA:

- TZA (highest) -> DCAEVT1 -> DCAEVT2 (lowest)
- TZAU (highest) -> DCAEVT1U -> DCAEVT2U (lowest)
- TZAD (highest) -> DCAEVT1D -> DCAEVT2D (lowest)

Output EPWMxB:

- TZB (highest) -> DCBEVT1 -> DCBEVT2 (lowest)
- TZBU (highest) -> DCBEVT1U -> DCBEVT2U (lowest)
- TZBD (highest) -> DCBEVT1D -> DCBEVT2D (lowest)

- **interrupt signal:**

DCAEVT1/2.interrupt signals generate trip zone interrupts to the PIE. To enable the interrupt, the user must set the DCAEVT1, DCAEVT2, DCBEVT1, or DCBEVT2 bits in the TZEINT register. Once one of these events occurs, an EPWMxTZINT interrupt is triggered, and the corresponding bit in the TZCLR register must be set in order to clear the interrupt.

- **soc signal:**

The DCAEVT1.soc signal interfaces with the event-trigger submodule and can be selected as an event which generates an ADC start-of-conversion-A (SOCA) pulse via the ETSEL[SOCASEL] bit. Likewise, the DCBEVT1.soc signal can be selected as an event which generates an ADC start-of-conversion-B (SOCB) pulse via the ETSEL[SOCBSEL] bit.

- **sync signal:**

The DCAEVT1.sync and DCBEVT1.sync events are ORed with the EPWMxSYNCl input signal and the TBCTL[SWFSYNC] signal to generate a synchronization pulse to the time-base counter.

The diagrams below show how the DCAEVT1, DCAEVT2 or DCEVTFILT signals are processed to generate the digital compare A event force, interrupt, soc and sync signals.

Figure 18-50. DCAEVT1 Event Triggering

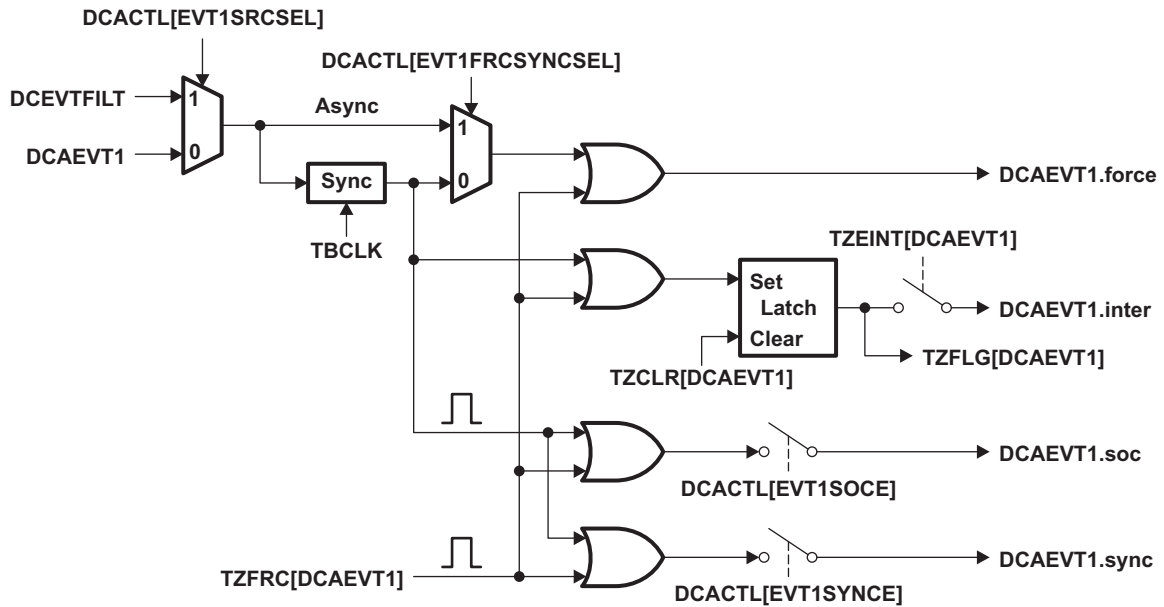


Figure 18-51. DCAEVT2 Event Triggering

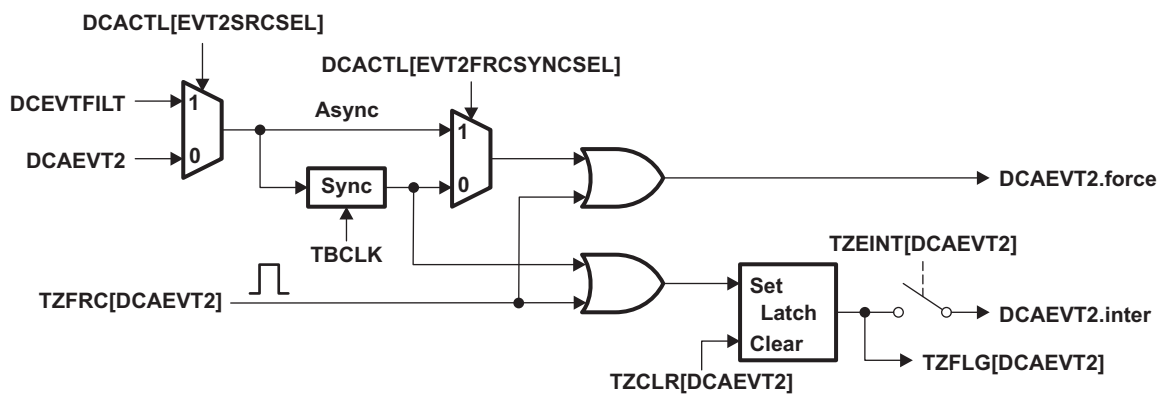


Figure 18-52 and Figure 18-53 show how the DCBEVT1, DCBEVT2 or DCEVTFLT signals are processed to generate the digital compare B event force, interrupt, soc and sync signals.

Figure 18-52. DCBEVT1 Event Triggering

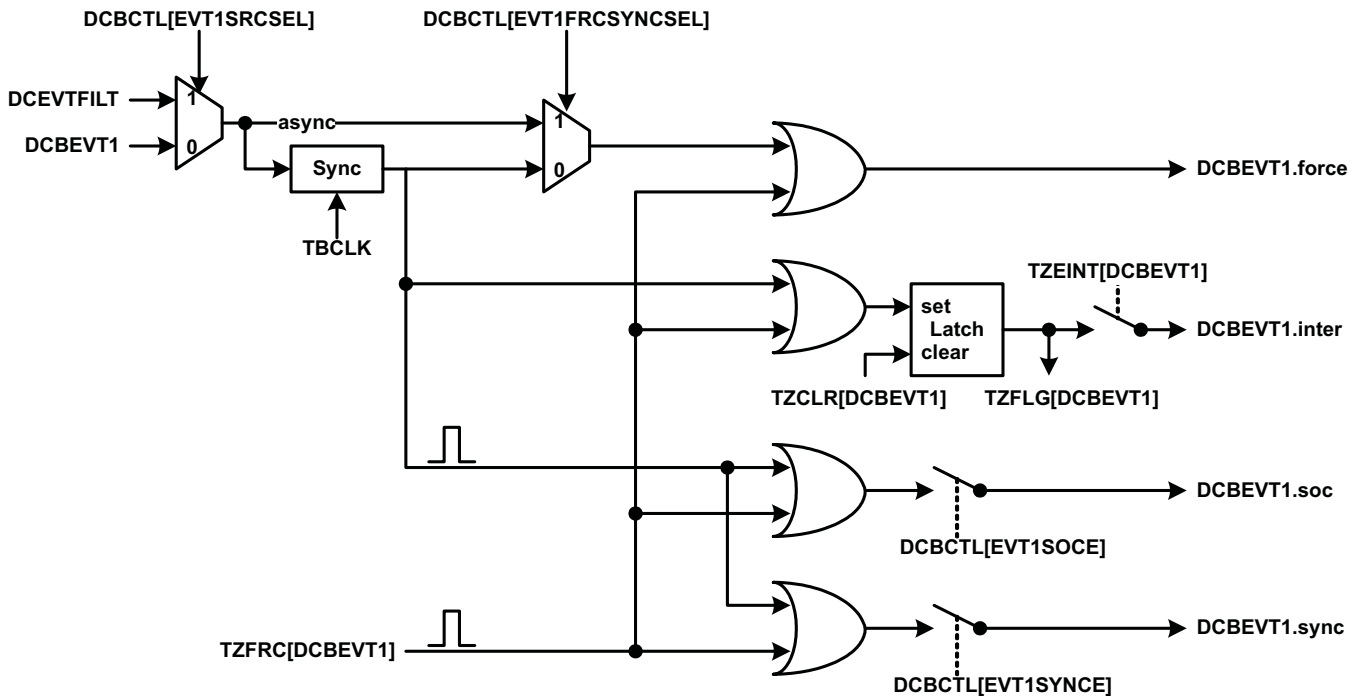
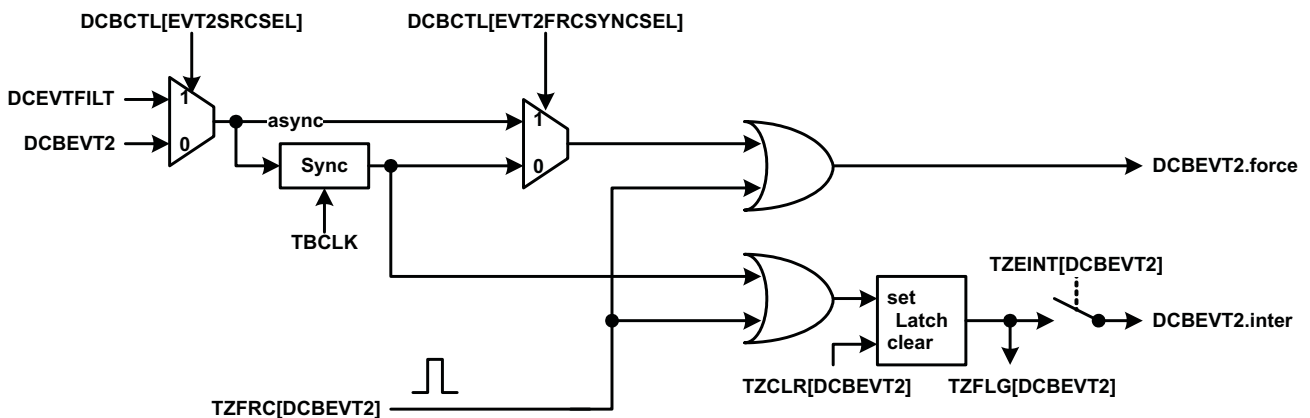


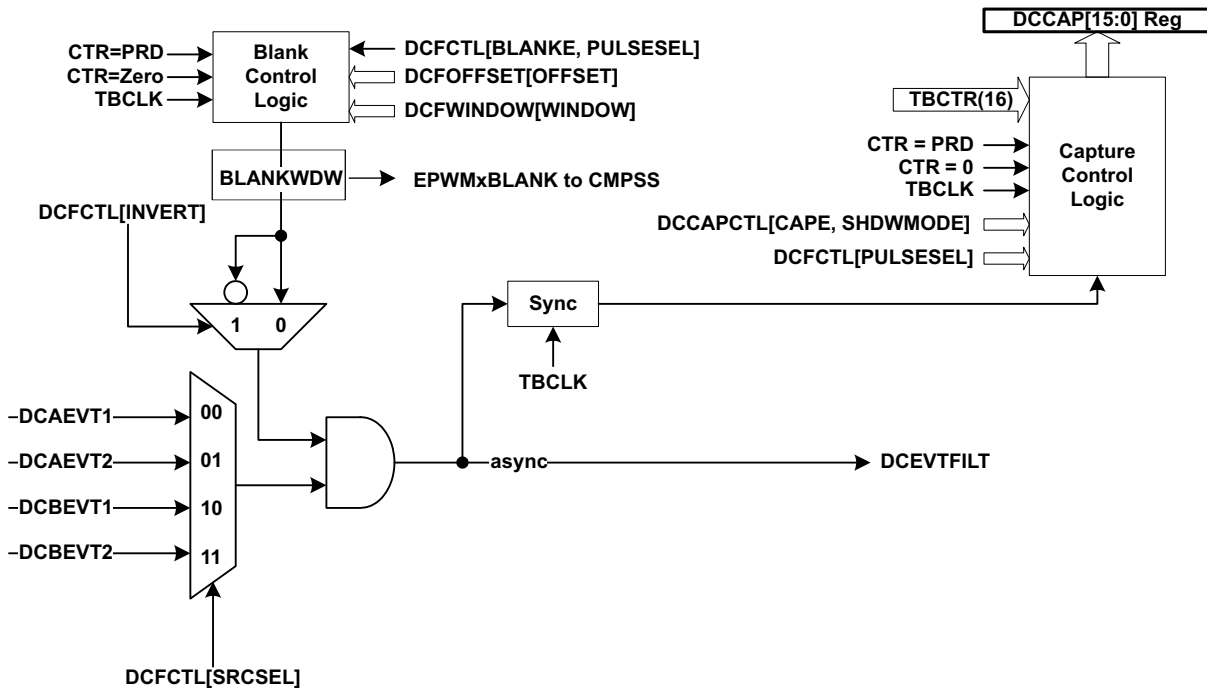
Figure 18-53. DCBEVT2 Event Triggering



#### 18.11.4.2 Event Filtering

The DCAEVT1/2 and DCBEVT1/2 events can be filtered via event filtering logic to remove noise by optionally blanking events for a certain period of time. This is useful for cases where the analog comparator outputs may be selected to trigger DCAEVT1/2 and DCBEVT1/2 events, and the blanking logic is used to filter out potential noise on the signal prior to tripping the PWM outputs or generating an interrupt or ADC start-of-conversion. The event filtering can also capture the TBCTR value of the trip event. The following figure shows the details of the event filtering logic.

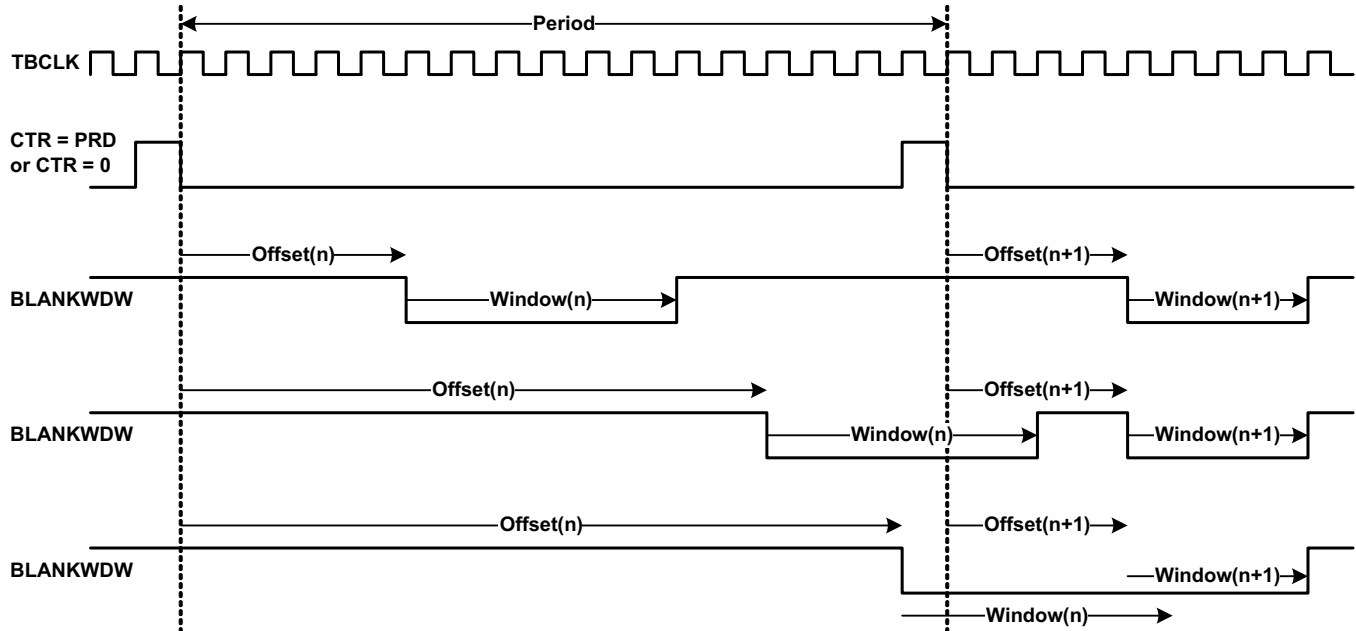
Figure 18-54. Event Filtering



If the blanking logic is enabled, one of the digital compare events – DCAEVT1, DCAEVT2, DCBEVT1, DCBEVT2 – is selected for filtering. The blanking window, which filters out all event occurrences on the signal while it is active, will be aligned to either a CTR = PRD pulse or a CTR = 0 pulse or both CTR = PRD and CTR = 0 (configured by the DCFCTL[PULSESEL] bits). An offset value in TBCLK counts is programmed into the DCFOFFSET register, which determines at what point after the CTR = PRD or CTR = 0 pulse the blanking window starts. The duration of the blanking window, in number of TBCLK counts after the offset counter expires, is written to the DCFWINDOW register by the application. During the blanking window, all events are ignored. Before and after the blanking window ends, events can generate soc, sync, interrupt, and force signals as before.

The diagram below illustrates several timing conditions for the offset and blanking window within an ePWM period. Notice that if the blanking window crosses the CTR = 0 or CTR = PRD boundary, the next window still starts at the same offset value after the CTR = 0 or CTR = PRD pulse.

Figure 18-55. Blanking Window Timing Diagram



### 18.11.4.3 Valley Switching

Event filtering depicts the valley switching function along with the event filtering logic described in [Section 18.11.4.2](#). This function can be used to achieve programmable valley switching without any additional external circuitry. This module provides an on-chip hardware mechanism that can:

- Capture the oscillation period
- Accurately delay the PWM switching instant
- Allow a programmable number of edges before the delay takes effect
- Provide multiple choices of triggers and events
- Allow easy adaptability for optimum performance under changing system/operating conditions

The DCxEV<sub>Ty</sub> signal needs further processing to support valley switching. Here is a brief description of how valley switching function is enabled:

1. Select one of the DCxEV<sub>Ty</sub> events as input to the valley switching block (DCFCTL[SRCSEL]) with an option to add the blanking window (Blank Control Logic). This is where the comparator output (or external input) above is selected as an input to the valley switching block.
2. Configure the edge filter to capture 'n' rising, falling or both edges through the edge selection logic (DCFCTL[EDGEMODE, EDGECOUNT]).
3. Select the correct event to reset and restart the edge filter (VCAPCTL[TRIGSEL]). Edge capturing event is triggered or armed by this selected edge.
4. Enable valley capture logic (VCAPCTL[VCAPE]).
5. Select the start edge that will indicate the start of capture for oscillation period measurement (VCNTCFG[STARTEDGE]). This is where the 16-bit counter starts counting.
6. Select the stop edge (VCNTCFG[STOPEDGE]) that will indicate the edge at which the 16-bit counter stops counting. The captured counter value (CNTVAL) provides oscillation period information.
  - The STOPEDGE value must always be greater than STARTEDGE value.
7. Configure and apply the captured delay (CNTVAL) to the edge filtered DCxEV<sub>Ty</sub> signal. The CNTVAL value may be applied as is or applied in conjunction with a software programmed value (useful for offset adjustment) (SWVDELVAL) or only a fraction of the delay may be applied with or without SWVDELVAL. This is useful to correctly apply a delay corresponding to the valley point. (VCAPCTL[VDELAYDIV])

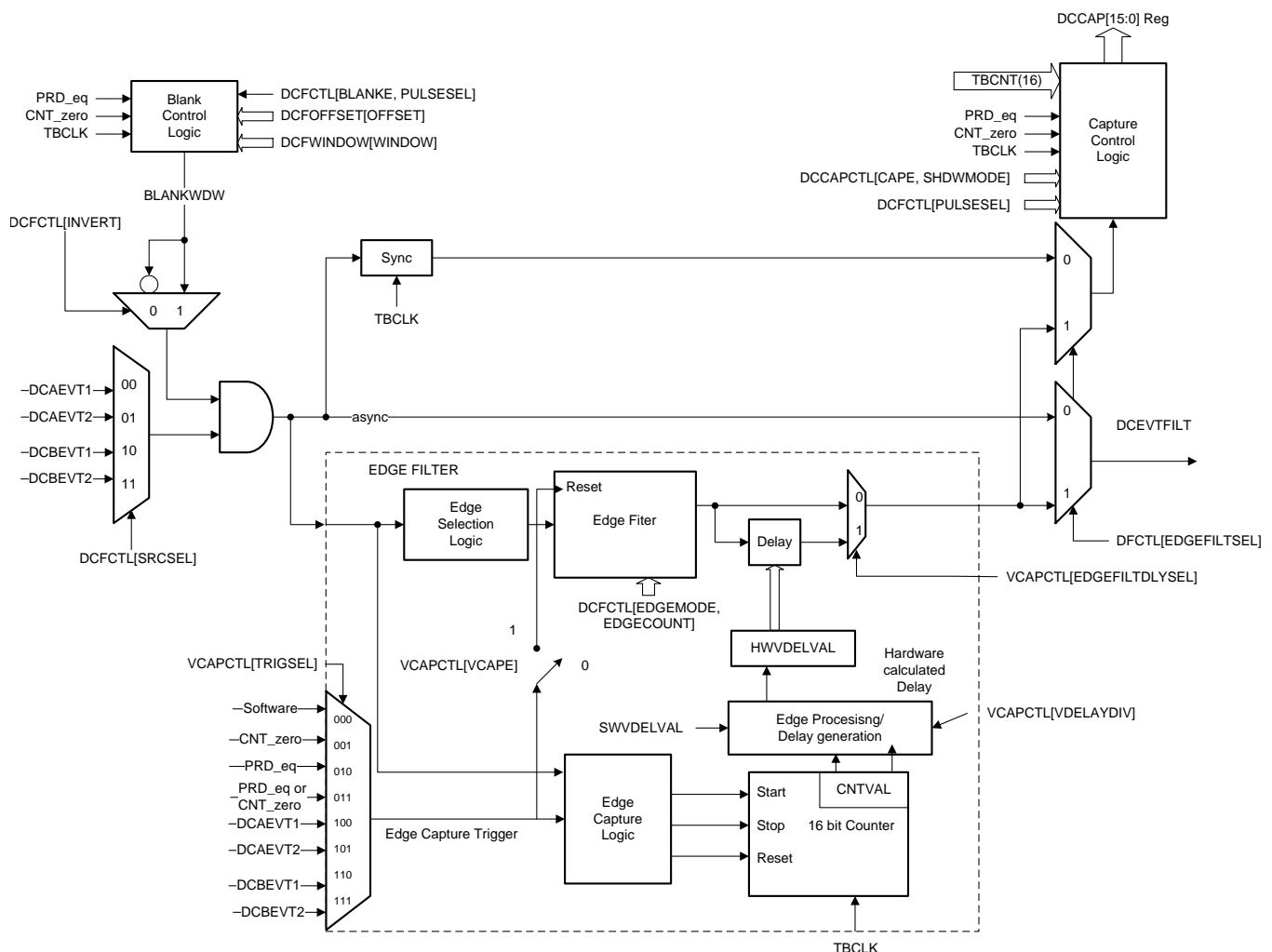
8. Configure VCAPCTL[EDGEFILTDLYSEL] to apply hardware delay based on the captured value above.

Once the counter is stopped, counter value is copied into CNTVAL register and counter is reset to zero. No further captures are done until the logic is triggered again by occurrence of event selected by VCAPCTL[TRIGSEL]. In this implementation, the software trigger is used as the source for VCAPCTL[TRIGSEL]. Upon occurrence of the trigger event, irrespective of the current status of the counter, the counter is reset and starts counting from zero upon occurrence of the STARTEDGE. Similarly, upon occurrence of the trigger event, the edge filter is reset and starts counting from zero upon occurrence of the STARTEDGE.

Output from the valley switching block (DCEVTFILT) is then used to synchronize the PWM time-base. The process is shown in the figure below.

**NOTE:** A specific application example showcasing the usage of valley switching hardware and software is available on the controlSuite.

Figure 18-56. Valley Switching



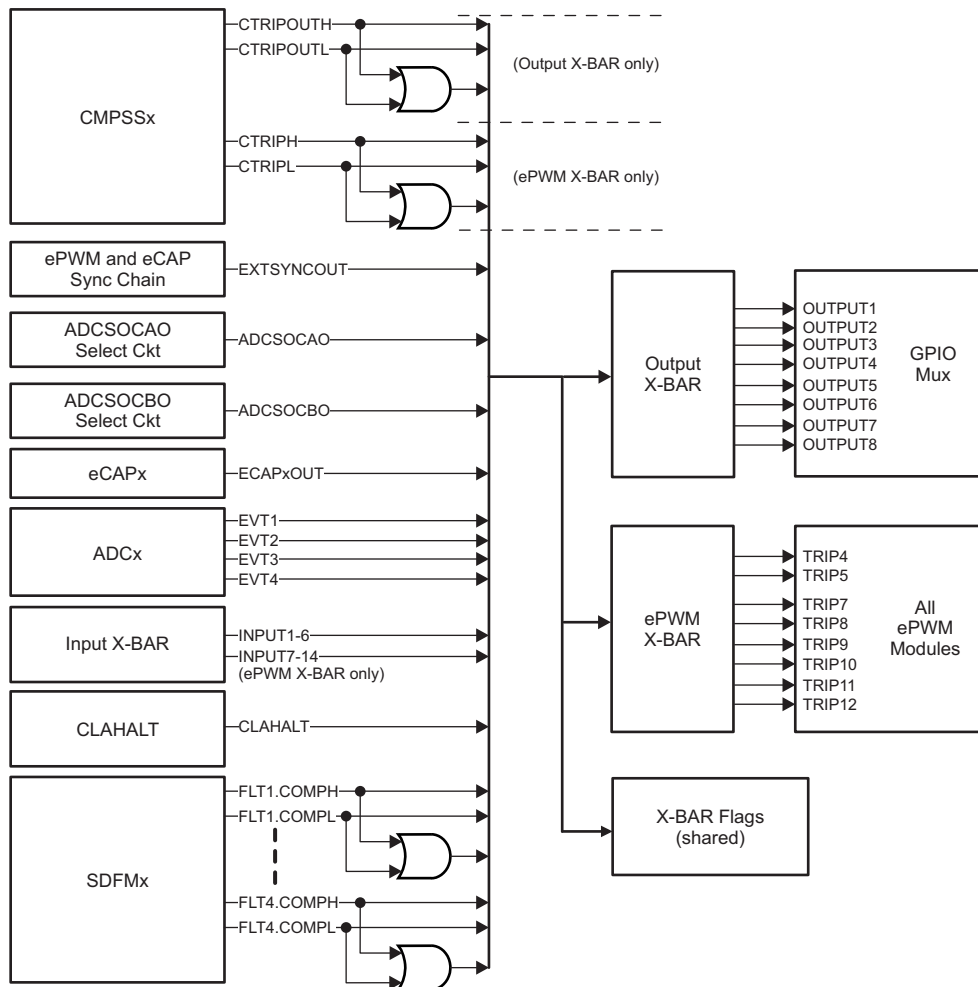
## 18.12 ePWM X-BAR

The figure below shows the architecture of the ePWM X-Bar. This module enables selection of various trigger sources into any of the eight dedicated ETWPM trips inputs, namely the TRIP4, TRIP5, TRIP7, TRIP8, TRIP9, TRIP10, TRIP11 and TRIP12.

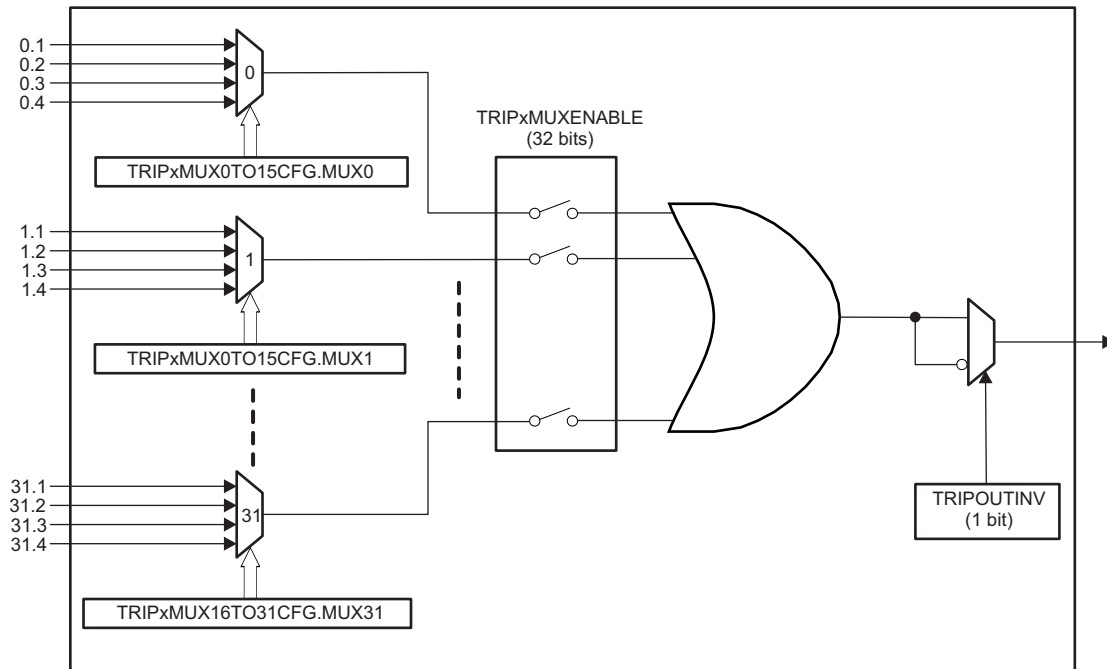


**NOTE:** Please refer to the X-BAR chapter for more information on the X-BAR modules, including X-BAR flags.

**Figure 18-57. ePWM X-BAR**



The ePWM X-BAR has eight outputs which are routed to each ePWM module. The following figure represents the architecture of a single output but it is identical to the architecture of all of the other outputs:

**Figure 18-58. ePWM XBAR Architecture - Single Output**


First, determine the signal(s) which should be passed to the ePWM by referencing the ePWM X-Bar Input Table. You may select up to one signal per mux (31 total muxes) for each TRIPx output. Select the inputs to each mux via the TRIPxMUX0TO15CFG and TRIPxMUX16TO31CFG registers. In order to pass any signal through to the EPWM, you must also enable the mux in the TRIPxMUXENABLE register. All muxes which are enabled will be logically OR'd before being passed on to the respective TRIPx signal on the EPWM. You may also optionally invert the signal via the TRIPxOUTINV register.

**Table 18-13. ePWM X-BAR Mux Configuration Table**

Mux	1	2	3	4
0	CMPSS1.CTRIPH	CMPSS1.CTRIPH_OR_CTRIPL	ADCAEVT1	ECAP1OUT
1	CMPSS1.CTRIPL	INPUTXBAR1		ADCCEVT1
2	CMPSS2.CTRIPH	CMPSS2.CTRIPH_OR_CTRIPL	ADCAEVT2	ECAP2OUT
3	CMPSS2.CTRIPL	INPUTXBAR2		ADCCEVT2
4	CMPSS3.CTRIPH	CMPSS3.CTRIPH_OR_CTRIPL	ADCAEVT3	ECAP3OUT
5	CMPSS3.CTRIPL	INPUTXBAR3		ADCCEVT3
6	CMPSS4.CTRIPH	CMPSS4.CTRIPH_OR_CTRIPL	ADCAEVT4	ECAP4OUT
7	CMPSS4.CTRIPL	INPUTXBAR4		ADCCEVT4
8	CMPSS5.CTRIPH	CMPSS5.CTRIPH_OR_CTRIPL	ADCBEVT1	ECAP5OUT
9	CMPSS5.CTRIPL	INPUTXBAR5		
10	CMPSS6.CTRIPH	CMPSS6.CTRIPH_OR_CTRIPL	ADCBEVT2	ECAP6OUT
11	CMPSS6.CTRIPL	INPUTXBAR6		
12	CMPSS7.CTRIPH	CMPSS7.CTRIPH_OR_CTRIPL	ADCBEVT3	ECAP7OUT
13	CMPSS7.CTRIPL	ADCSOAO		
14			ADCBEVT4	EXTSYNCOUT
15		ADCSOABO		
16	SD1FLT1.COMPH	SD1FLT1.COMPH_OR_COMPL		
17	SD1FLT1.COMPL	INPUT7		CLAHALT
18	SD1FLT2.COMPH	SD1FLT2.COMPH_OR_COMPL		
19	SD1FLT2.COMPL	INPUT8		
20	SD1FLT3.COMPH	SD1FLT3.COMPH_OR_COMPL		
21	SD1FLT3.COMPL	INPUT9		
22	SD1FLT4.COMPH	SD1FLT4.COMPH_OR_COMPL		
23		INPUT10		
24				
25		INPUT11		
26				
27		INPUT12		
28				
29		INPUT13		
30				
31		INPUT14		

**Note:** All unused and reserved positions are tied to '0'.

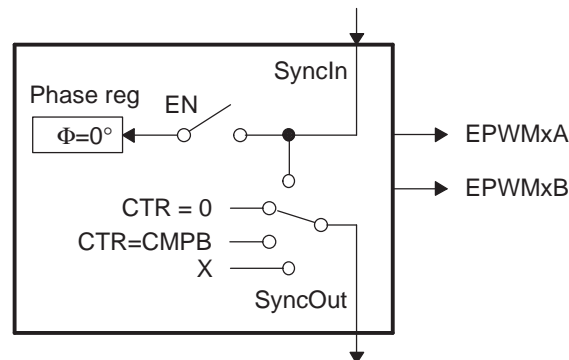
## 18.13 Applications to Power Topologies

An ePWM module has all the local resources necessary to operate completely as a standalone module or to operate in synchronization with other identical ePWM modules.

### 18.13.1 Overview of Multiple Modules

Previously in this chapter, all discussions have described the operation of a single module. To facilitate the understanding of multiple modules working together in a system, the ePWM module described in reference is represented by the more simplified block diagram shown in [Figure 18-59](#). This simplified ePWM block shows only the key resources needed to explain how a multiswitch power topology is controlled with multiple ePWM modules working together.

**Figure 18-59. Simplified ePWM Module**



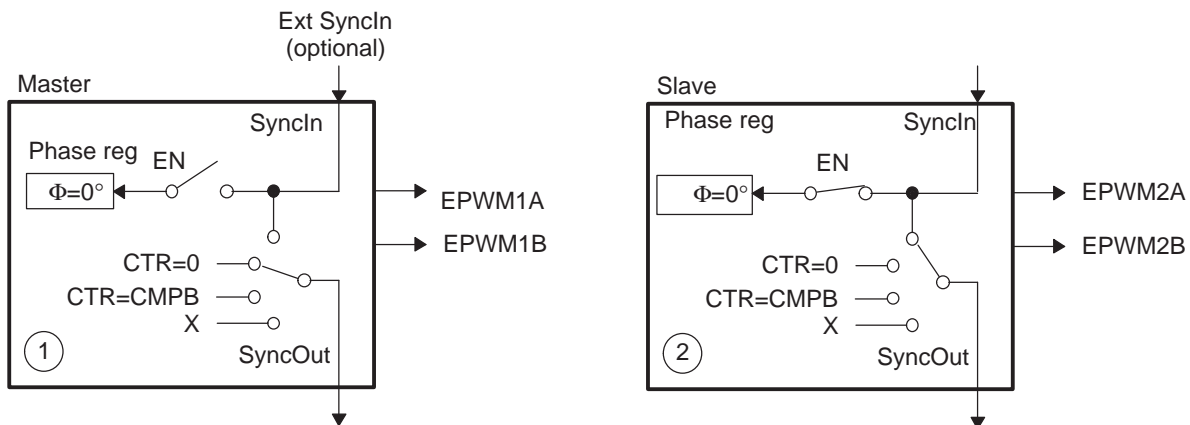
### 18.13.2 Key Configuration Capabilities

The key configuration choices available to each module are as follows:

- Options for SyncIn
  - Load own counter with phase register on an incoming sync strobe—enable (EN) switch closed
  - Do nothing or ignore incoming sync strobe—enable switch open
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)
- Options for SyncOut
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)

For each choice of SyncOut, a module may also choose to load its own counter with a new phase value on a SyncIn strobe input or choose to ignore it (that is, via the enable switch). Although various combinations are possible, the two most common—master module and slave module modes—are shown in [Figure 18-60](#).

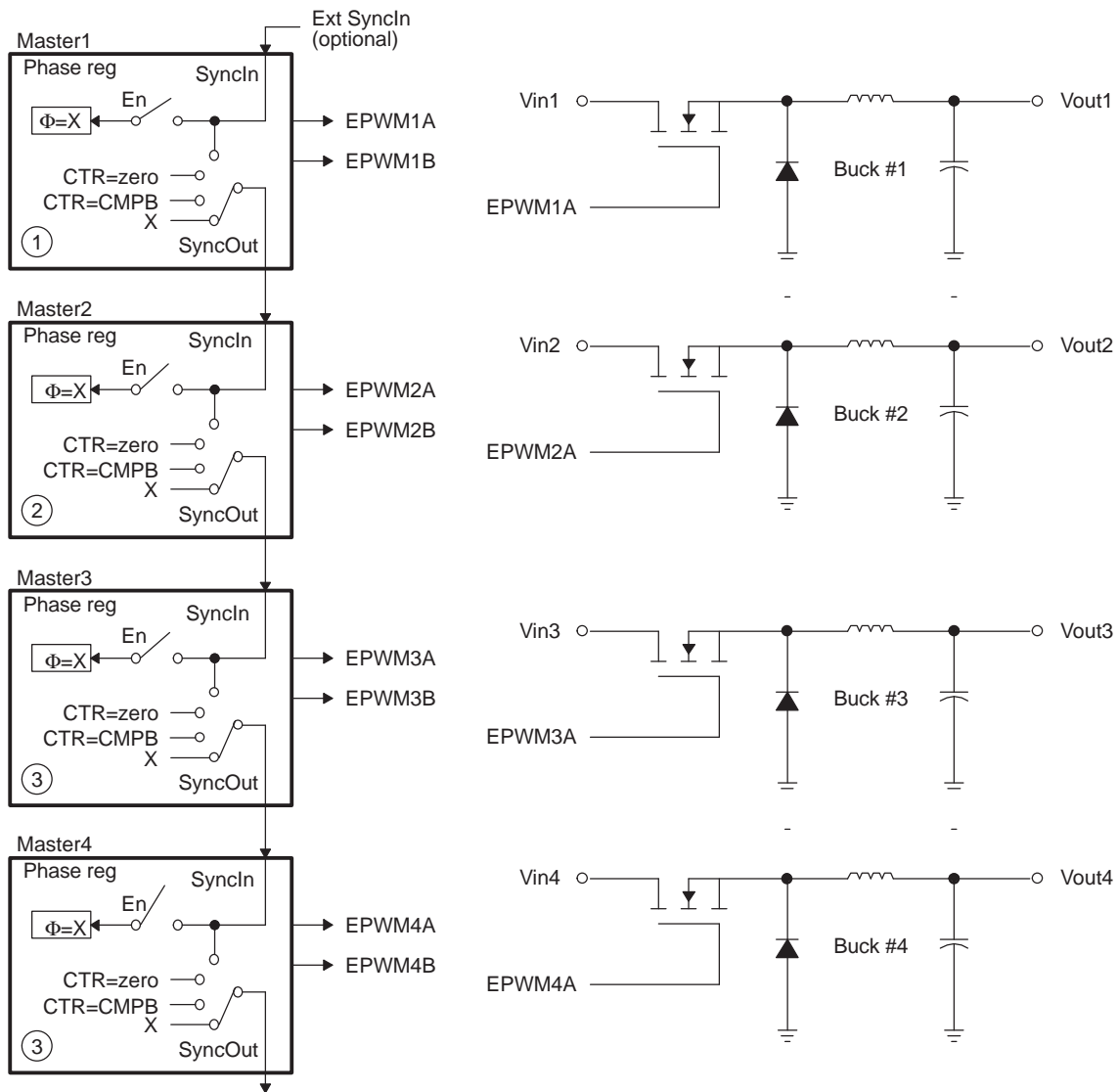
Figure 18-60. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave



### 18.13.3 Controlling Multiple Buck Converters With Independent Frequencies

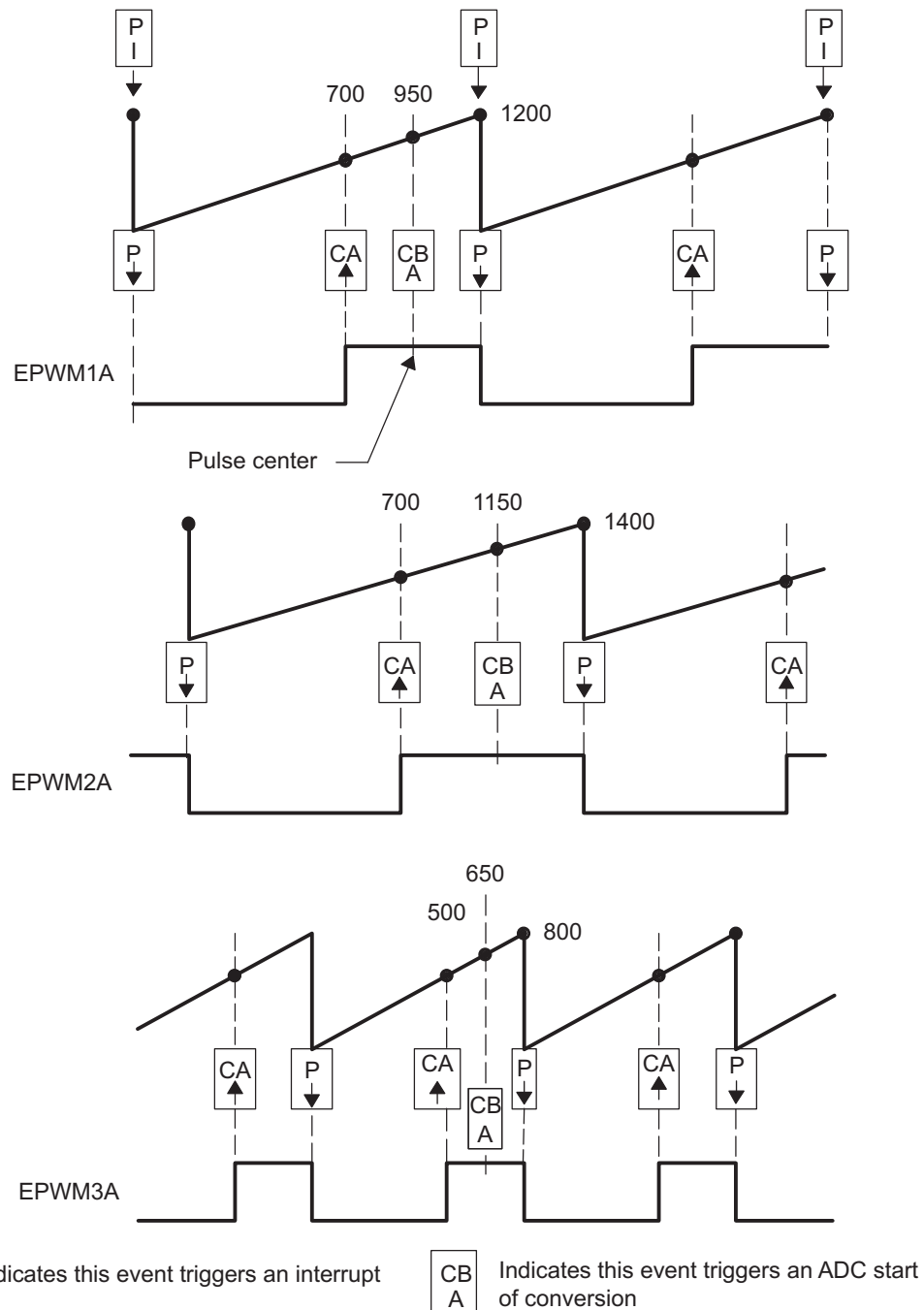
One of the simplest power converter topologies is the buck. A single ePWM module configured as a master can control two buck stages with the same PWM frequency. If independent frequency control is required for each buck converter, then one ePWM module must be allocated for each converter stage. Figure 18-61 shows four buck stages, each running at independent frequencies. In this case, all four ePWM modules are configured as Masters and no synchronization is used. Figure 18-62 shows the waveforms generated by the setup shown in Figure 18-61; note that only three waveforms are shown, although there are four stages.

Figure 18-61. Control of Four Buck Stages. Here  $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$



NOTE:  $\phi = X$  indicates value in phase register is a "don't care"

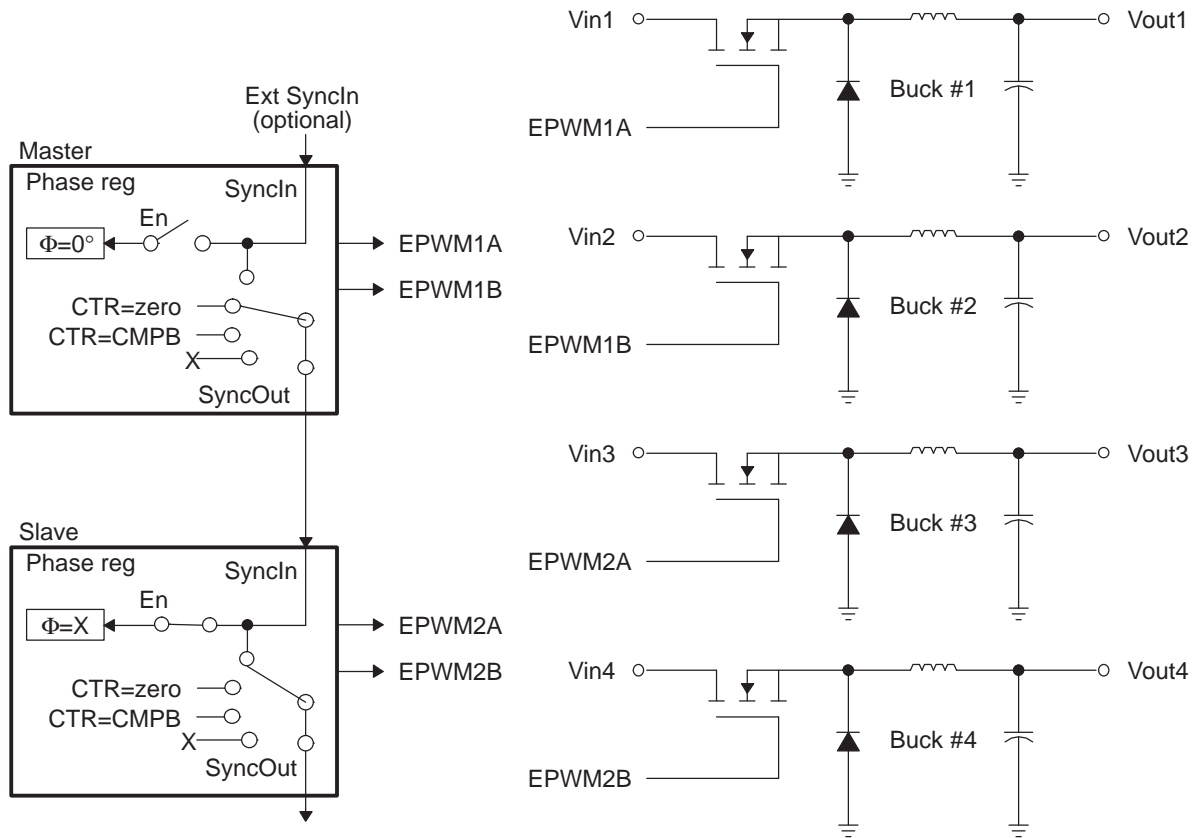
Figure 18-62. Buck Waveforms for Figure 18-61 (Note: Only three bucks shown here)



### 18.13.4 Controlling Multiple Buck Converters With Same Frequencies

If synchronization is a requirement, ePWM module 2 can be configured as a slave and can operate at integer multiple (N) frequencies of module 1. The sync signal from master to slave ensures these modules remain locked. Figure 18-63 shows such a configuration; Figure 18-64 shows the waveforms generated by the configuration.

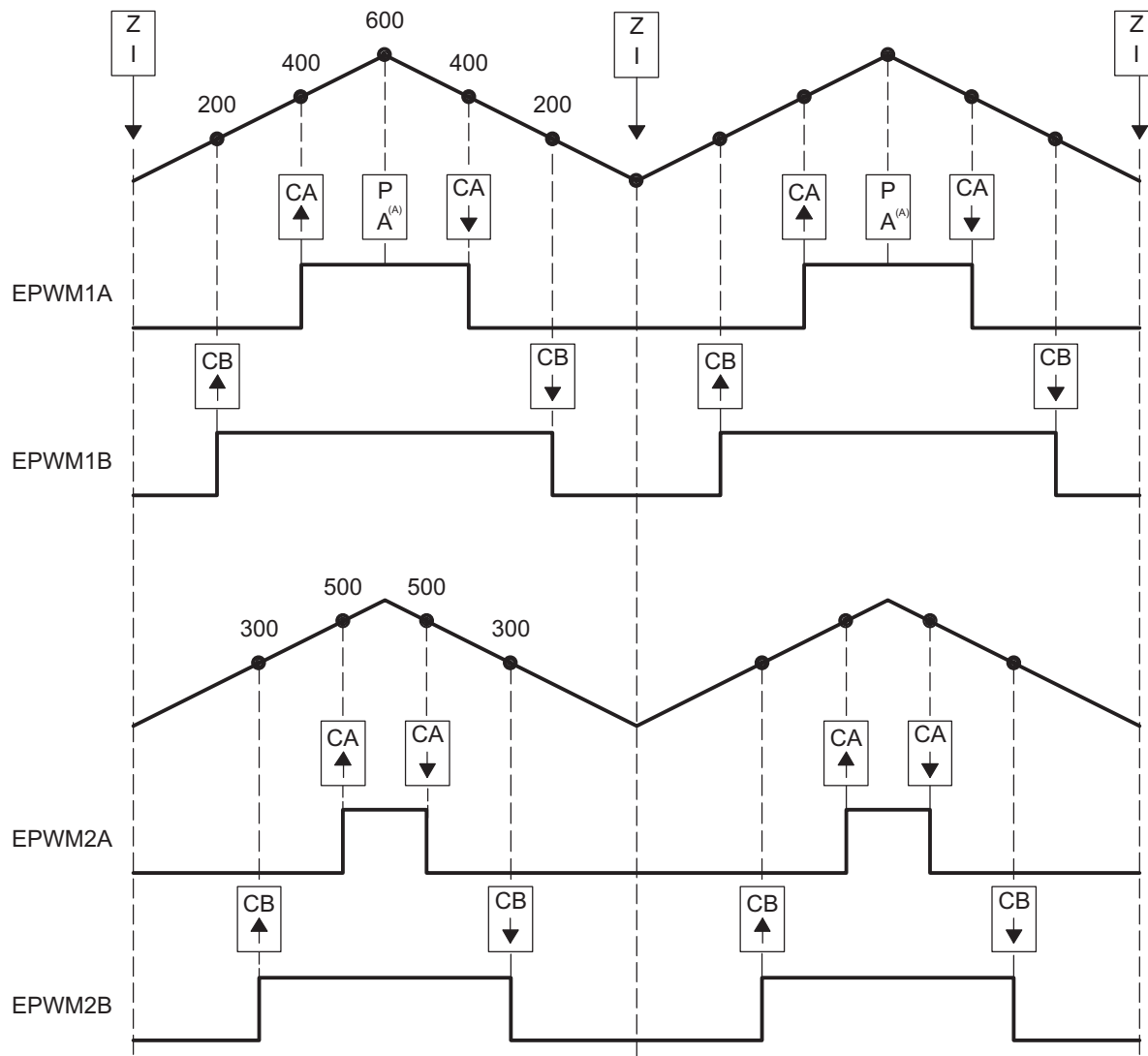
Figure 18-63. Control of Four Buck Stages. (Note:  $F_{PWM2} = N \times F_{PWM1}$ )



NOTE:  $\phi = X$  indicates value in phase register is a "don't care"



Figure 18-64. Buck Waveforms for Figure 18-63 (Note:  $F_{PWM2} = F_{PWM1}$ )



A Starts ADC conversion.

### 18.13.5 Controlling Multiple Half H-Bridge (HHB) Converters

Topologies that require control of multiple switching elements can also be addressed with these same ePWM modules. It is possible to control a Half-H bridge stage with a single ePWM module. This control can be extended to multiple stages. Figure 18-65 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 18-66 shows the waveforms generated by the configuration shown in Figure 18-65.

Module 2 (slave) is configured for Sync flow-through; if required, this configuration allows for a third Half-H bridge to be controlled by PWM module 3 and also, most importantly, to remain in synchronization with master module 1.

Figure 18-65. Control of Two Half-H Bridge Stages ( $F_{P_{WM2}} = N \times F_{P_{WM1}}$ )

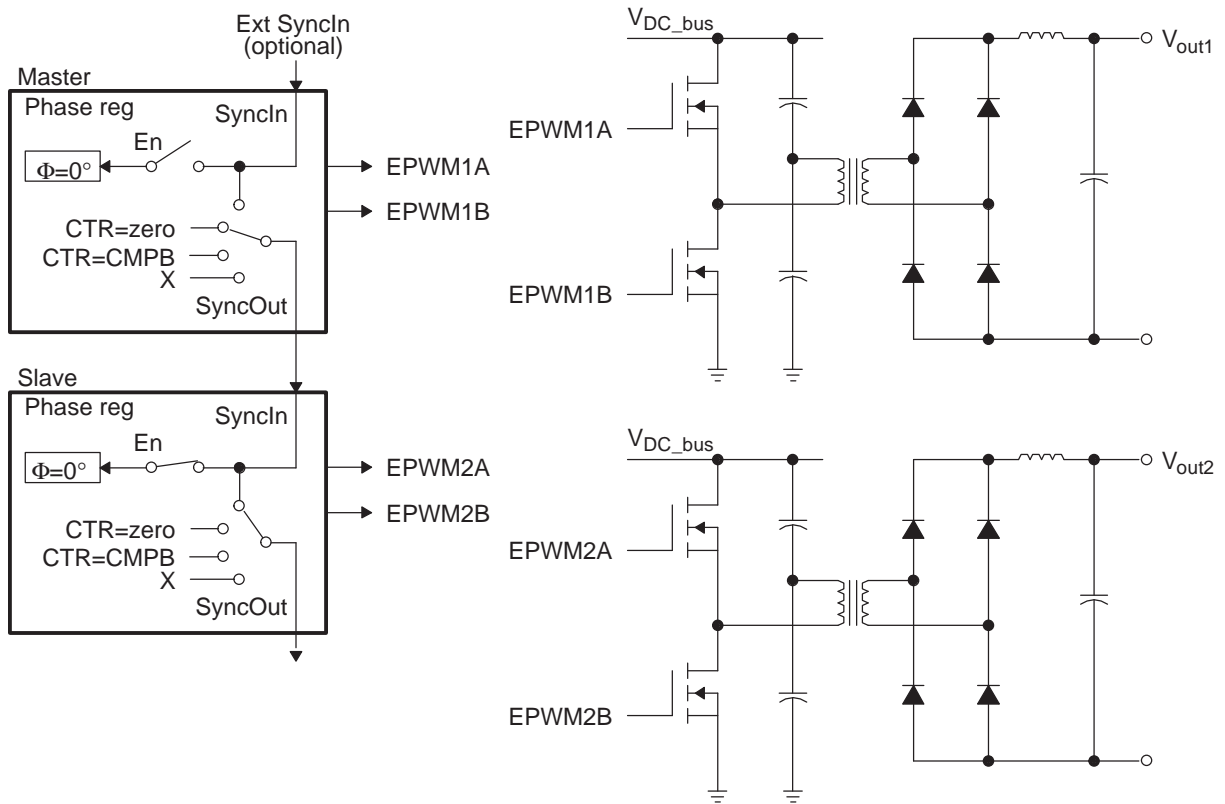
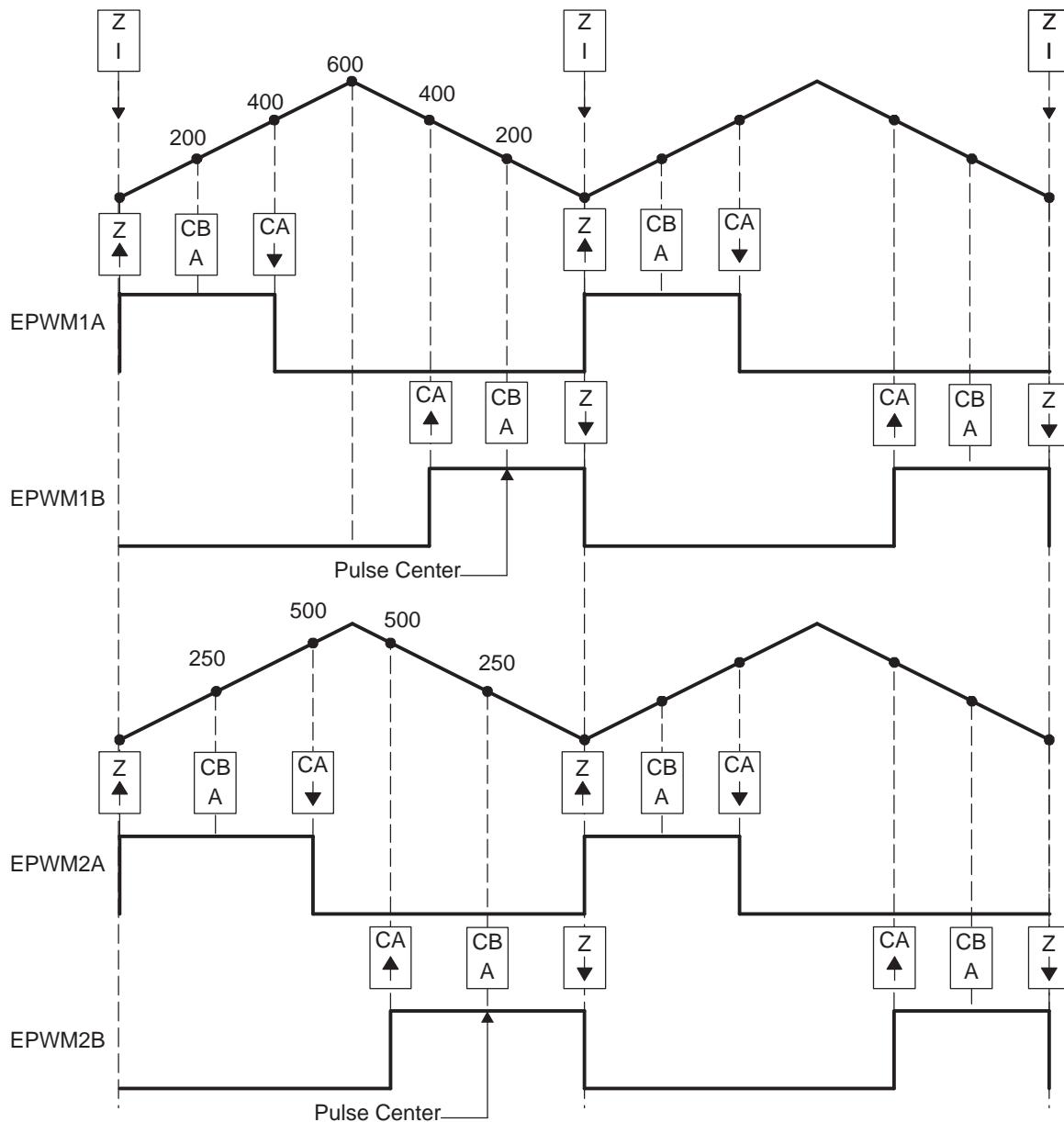


Figure 18-66. Half-H Bridge Waveforms for Figure 18-65 (Note: Here  $F_{PWM2} = F_{PWM1}$ )



### 18.13.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM)

The idea of multiple modules controlling a single power stage can be extended to the 3-phase Inverter case. In such a case, six switching elements can be controlled using three PWM modules, one for each leg of the inverter. Each leg must switch at the same frequency and all legs must be synchronized. A master + two slaves configuration can easily address this requirement. Figure 18-67 shows how six PWM modules can control two independent 3-phase Inverters; each running a motor.

As in the cases shown in the previous sections, we have a choice of running each inverter at a different frequency (module 1 and module 4 are masters as in Figure 18-67), or both inverters can be synchronized by using one master (module 1) and five slaves. In this case, the frequency of modules 4, 5, and 6 (all equal) can be integer multiples of the frequency for modules 1, 2, 3 (also all equal).

Figure 18-67. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control

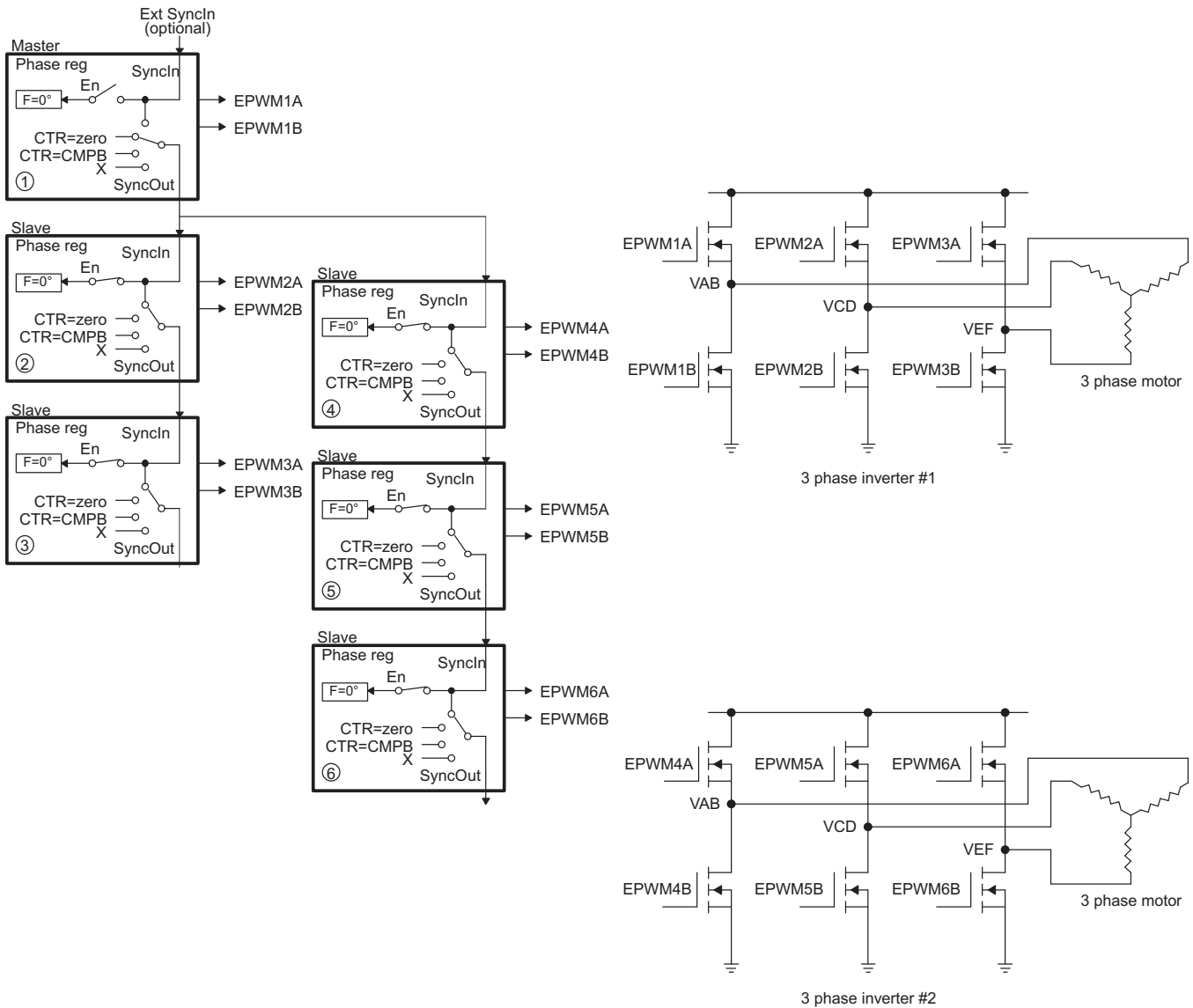
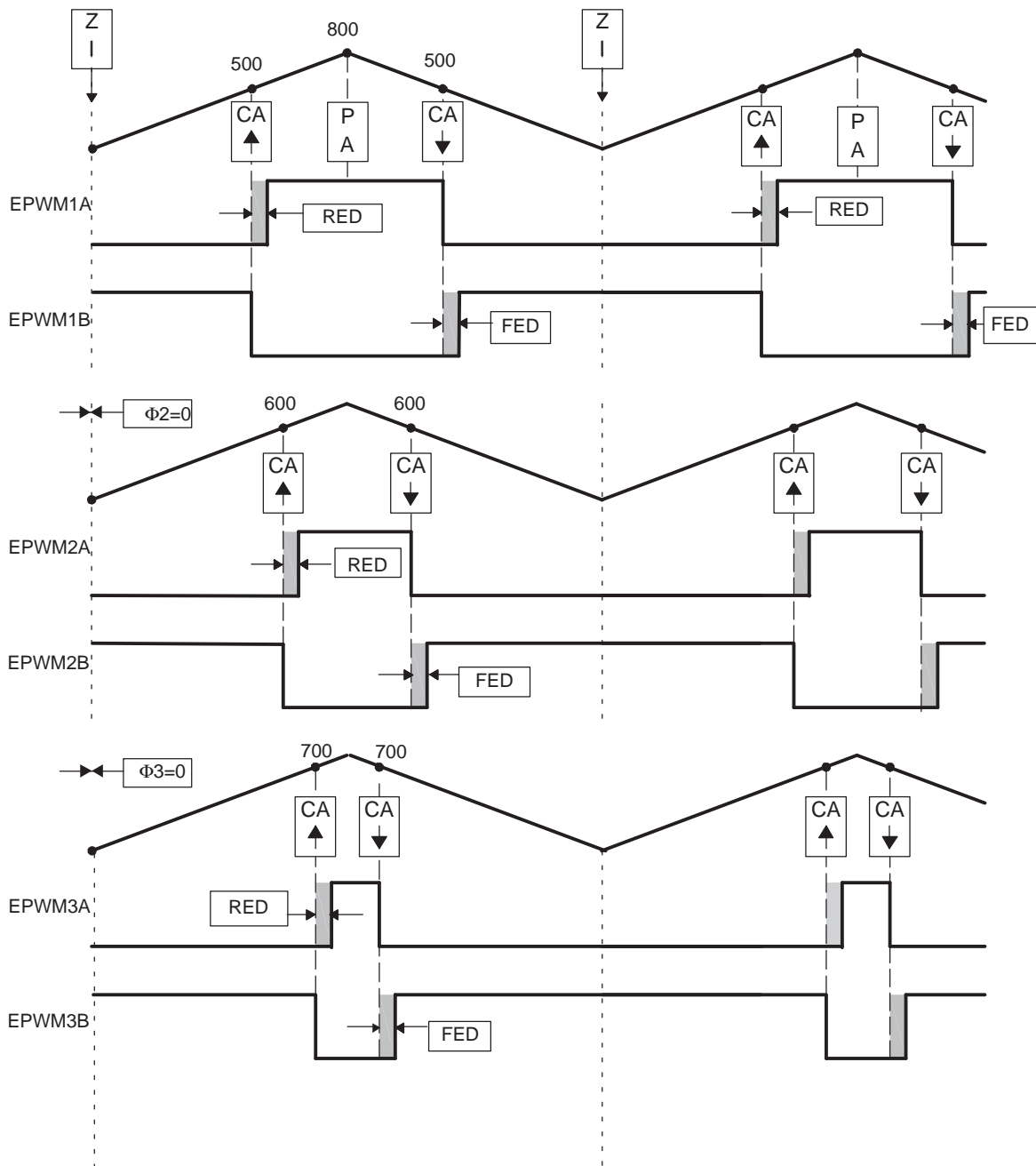


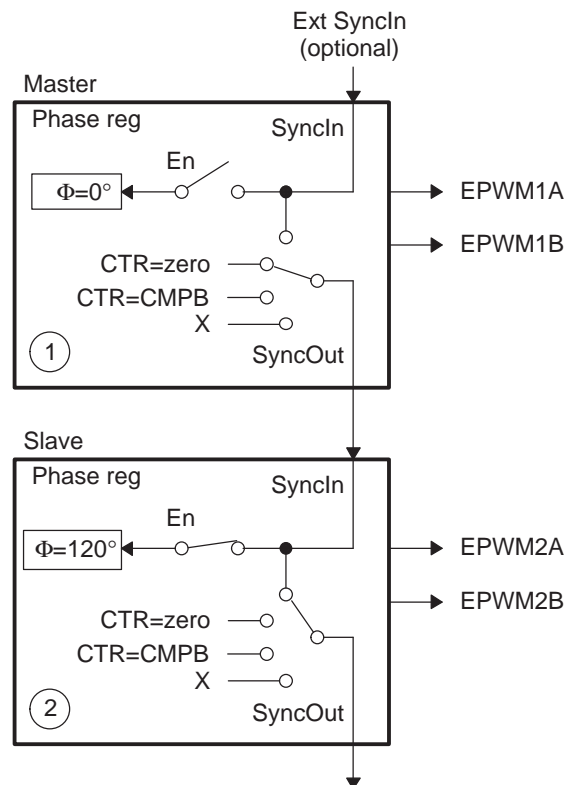
Figure 18-68. 3-Phase Inverter Waveforms for Figure 18-67 (Only One Inverter Shown)



### 18.13.7 Practical Applications Using Phase Control Between PWM Modules

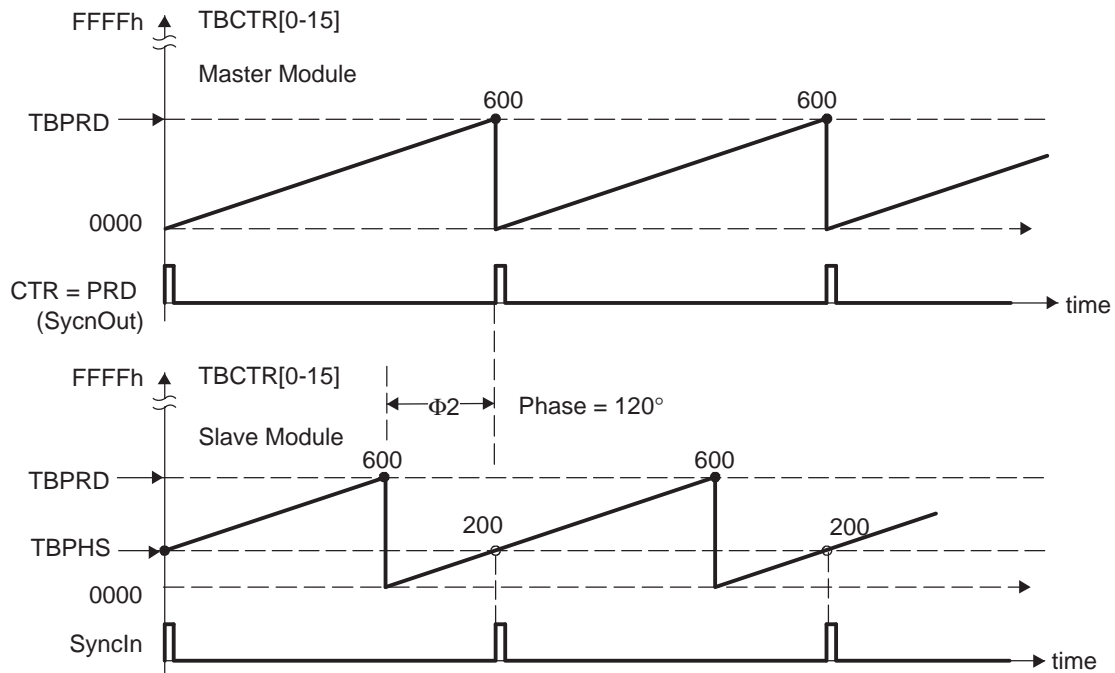
So far, none of the examples have made use of the phase register (TBPHS). It has either been set to zero or its value has been a don't care. However, by programming appropriate values into TBPHS, multiple PWM modules can address another class of power topologies that rely on phase relationship between legs (or stages) for correct operation. As described in the TB module section, a PWM module can be configured to allow a Syncln pulse to cause the TBPHS register to be loaded into the TBCTR register. To illustrate this concept, the following figure shows a master and slave module with a phase relationship of  $120^\circ$  (that is, the slave leads the master).

**Figure 18-69. Configuring Two PWM Modules for Phase Control**



The following figure shows the associated timing waveforms for this configuration. Here, TBPRD = 600 for both master and slave. For the slave, TBPHS = 200 (that is,  $200/600 \times 360^\circ = 120^\circ$ ). Whenever the master generates a Syncln pulse (CTR = PRD), the value of TBPHS = 200 is loaded into the slave TBCTR register so the slave time-base is always leading the master's time-base by  $120^\circ$ .

**Figure 18-70. Timing Waveforms Associated With Phase Control Between Two Modules**



### 18.13.8 Controlling a 3-Phase Interleaved DC/DC Converter

A popular power topology that makes use of phase-offset between modules is shown in [Figure 18-71](#). This system uses three PWM modules, with module 1 configured as the master. To work, the phase relationship between adjacent modules must be  $F = 120^\circ$ . This is achieved by setting the slave TBPHS registers 2 and 3 with values of 1/3 and 2/3 of the period value, respectively. For example, if the period register is loaded with a value of 600 counts, then TBPHS (slave 2) = 200 and TBPHS (slave 3) = 400. Both slave modules are synchronized to the master 1 module.

This concept can be extended to four or more phases, by setting the TBPHS values appropriately. The following formula gives the TBPHS values for N phases:

$$TBPHS(N,M) = (TBPRD/N) \times (M-1)$$

Where:

N = number of phases

M = PWM module number

For example, for the 3-phase case (N=3), TBPRD = 600,

$$TBPHS(3,2) = (600/3) \times (2-1) = 200 \text{ (that is, Phase value for Slave module 2)}$$

$$TBPHS(3,3) = 400 \text{ (that is, Phase value for Slave module 3)}$$

[Figure 18-72](#) shows the waveforms for the configuration in [Figure 18-71](#).

Figure 18-71. Control of a 3-Phase Interleaved DC/DC Converter

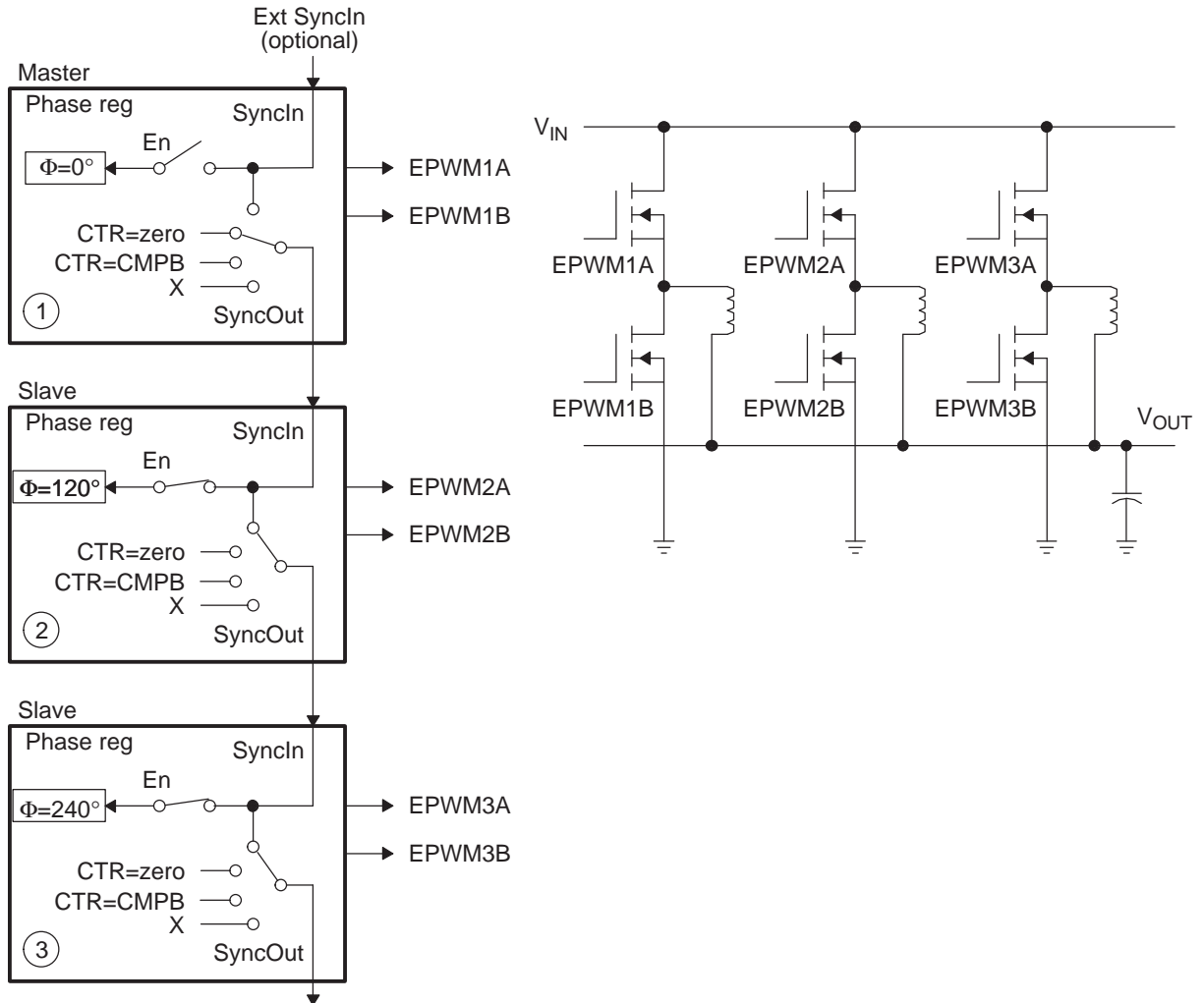
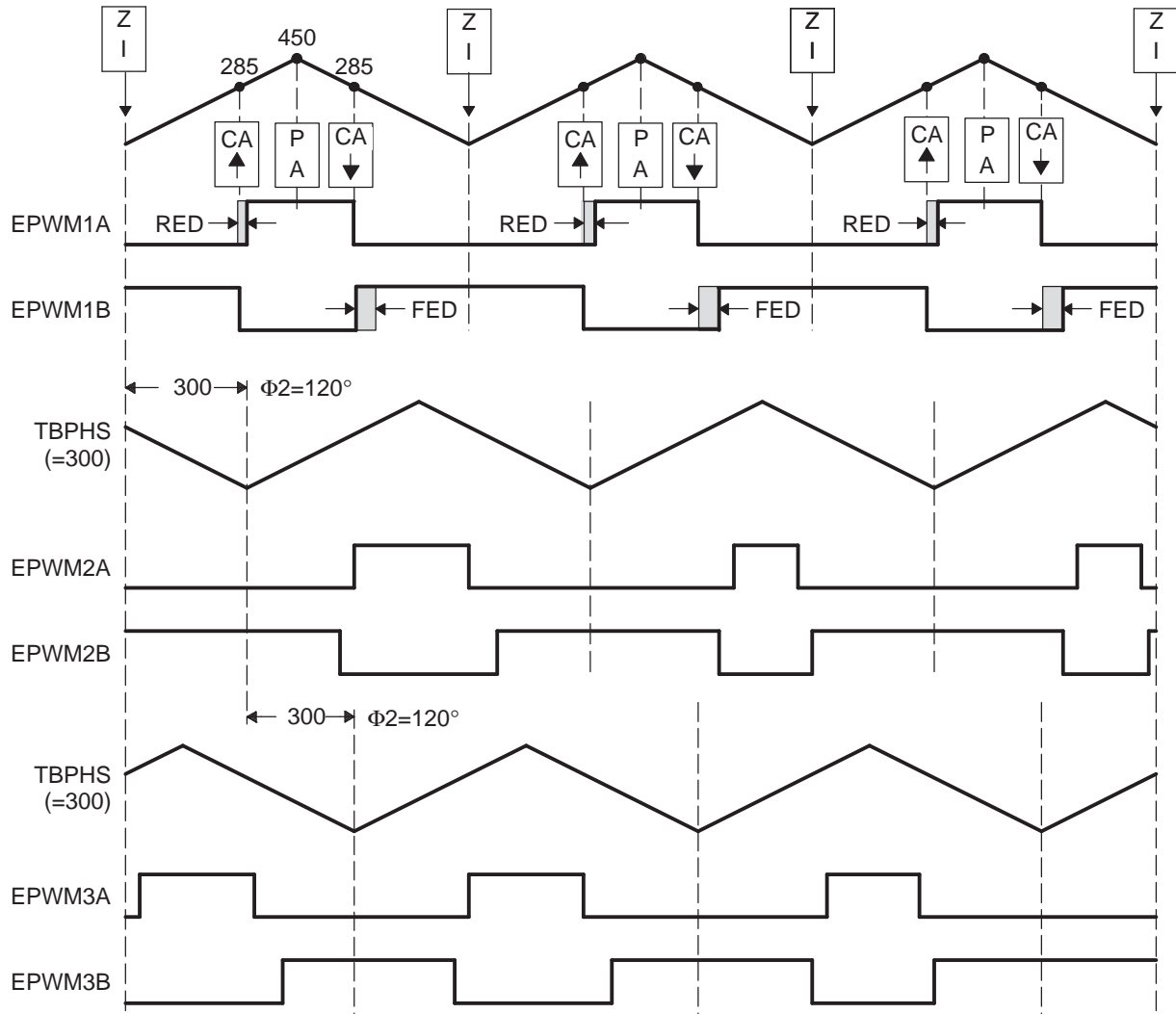




Figure 18-72. 3-Phase Interleaved DC/DC Converter Waveforms for Figure 18-71



### 18.13.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

The example given in the figure below assumes a static or constant phase relationship between legs (modules). In such a case, control is achieved by modulating the duty cycle. It is also possible to dynamically change the phase value on a cycle-by-cycle basis. This feature lends itself to controlling a class of power topologies known as *phase-shifted full bridge*, or *zero voltage switched full bridge*. Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead it is the phase relationship between legs. Such a system can be implemented by allocating the resources of two PWM modules to control a single power stage, which in turn requires control of four switching elements. [Figure 18-74](#) shows a master/slave module combination synchronized together to control a full H-bridge. In this case, both master and slave modules are required to switch at the same PWM frequency. The phase is controlled by using the slave's phase register (TBPHS). The master's phase register is not used and therefore can be initialized to zero.

**Figure 18-73. Controlling a Full-H Bridge Stage ( $F_{PWM2} = F_{PWM1}$ )**

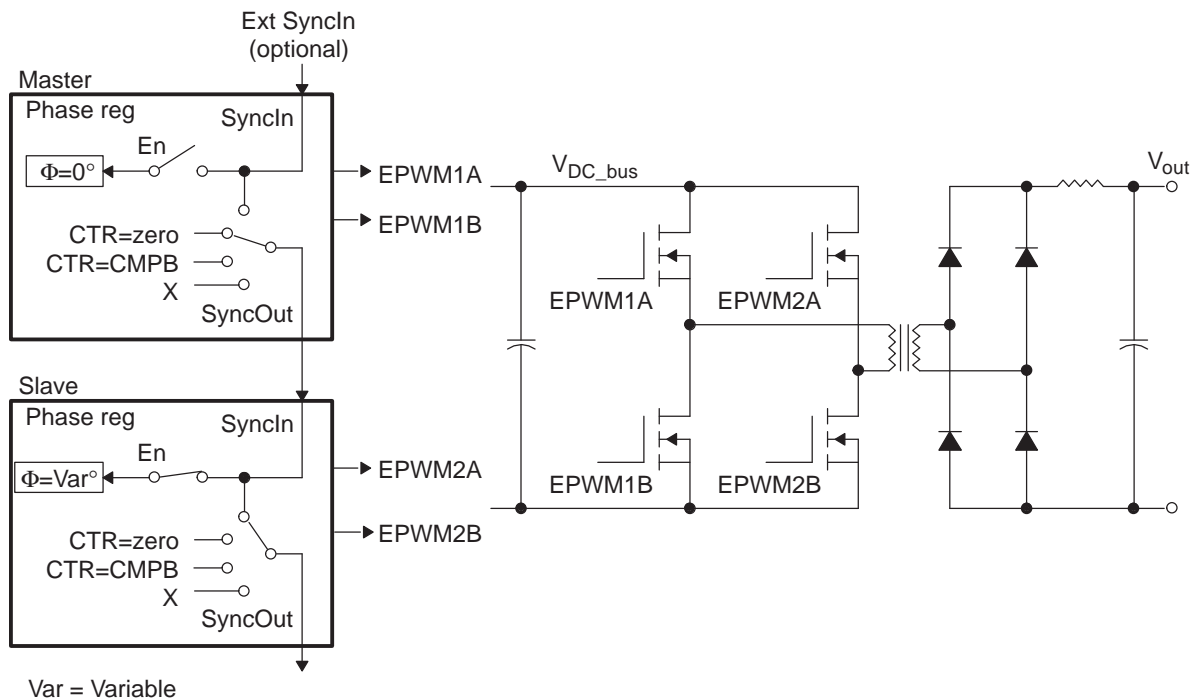
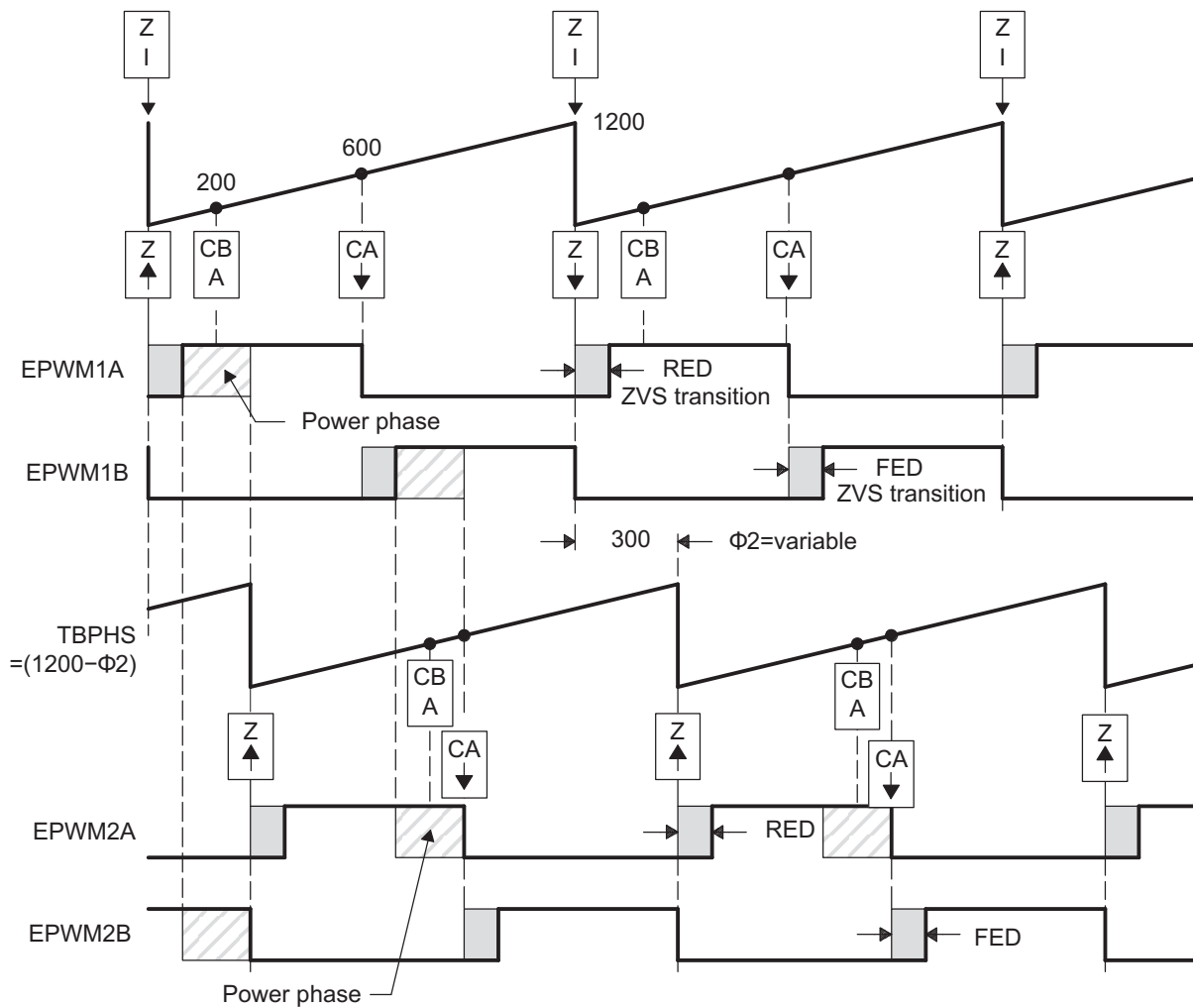
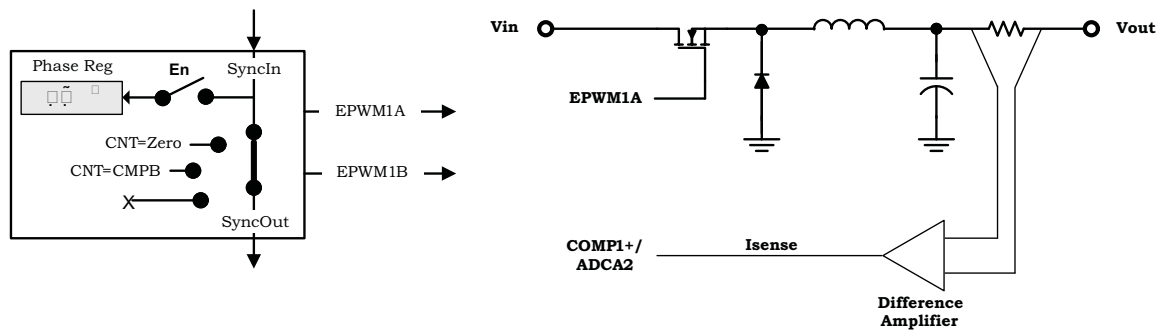
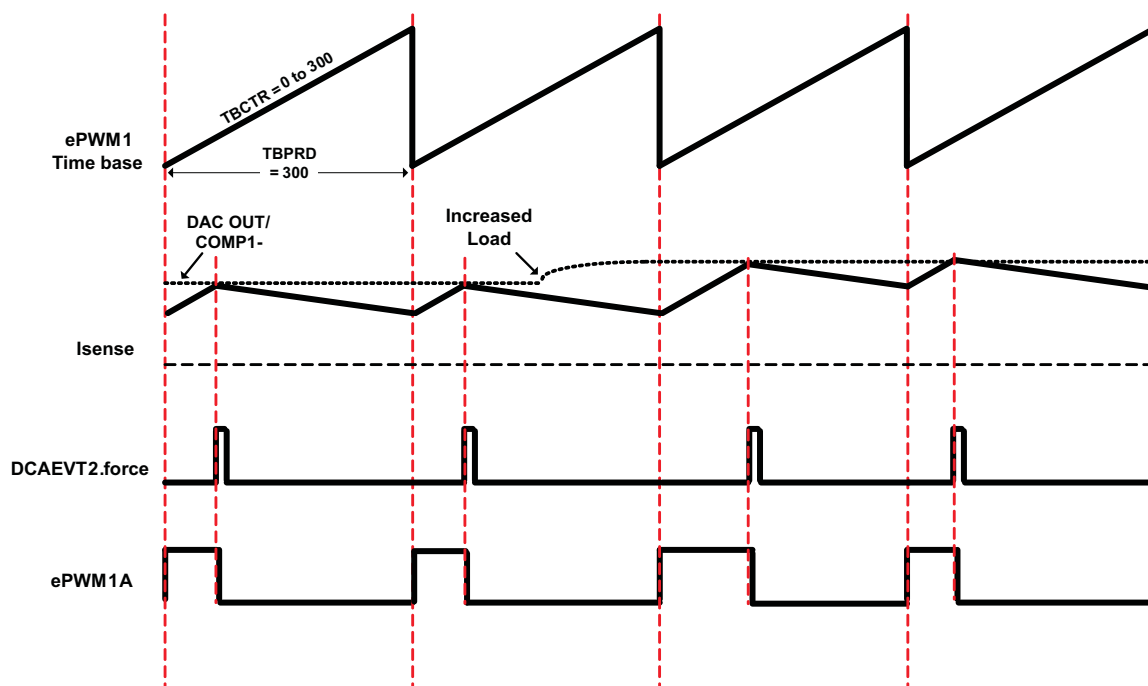


Figure 18-74. ZVS Full-H Bridge Waveforms



### 18.13.10 Controlling a Peak Current Mode Controlled Buck Module

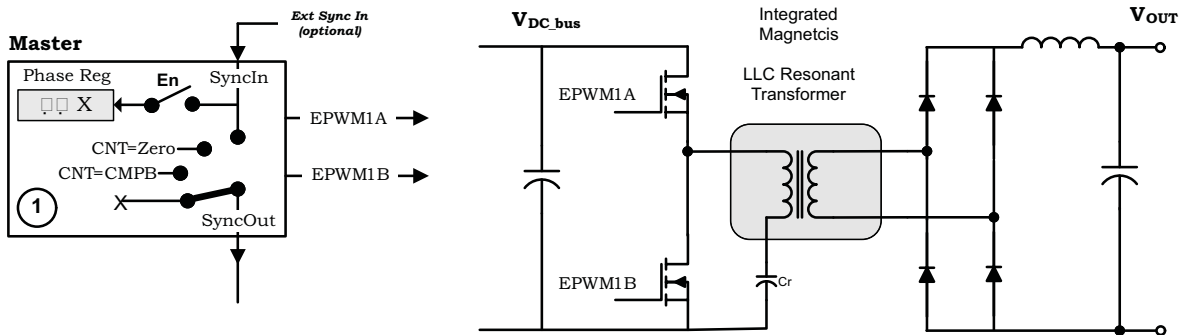
Peak current control techniques offer a number of benefits like automatic over current limiting, fast correction for input voltage variations and reducing magnetic saturation. Figure 18-75 shows the use of ePWM1A along with the on-chip analog comparator for buck converter topology. The output current is sensed through a current sense resistor and fed to the positive terminal of the on-chip comparator. The internal programmable 10-bit DAC can be used to provide a reference peak current at the negative terminal of the comparator. Alternatively, an external reference could be connected at this input. The comparator output is an input to the Digital compare sub-module. The ePWM module is configured in such a way so as to trip the ePWM1A output as soon as the sensed current reaches the peak reference value. A cycle-by-cycle trip mechanism is used. Figure 18-76 shows the waveforms generated by the configuration.

**Figure 18-75. Peak Current Mode Control of a Buck Converter**

**Figure 18-76. Peak Current Mode Control Waveforms for Figure 18-75**


### 18.13.11 Controlling H-Bridge LLC Resonant Converter

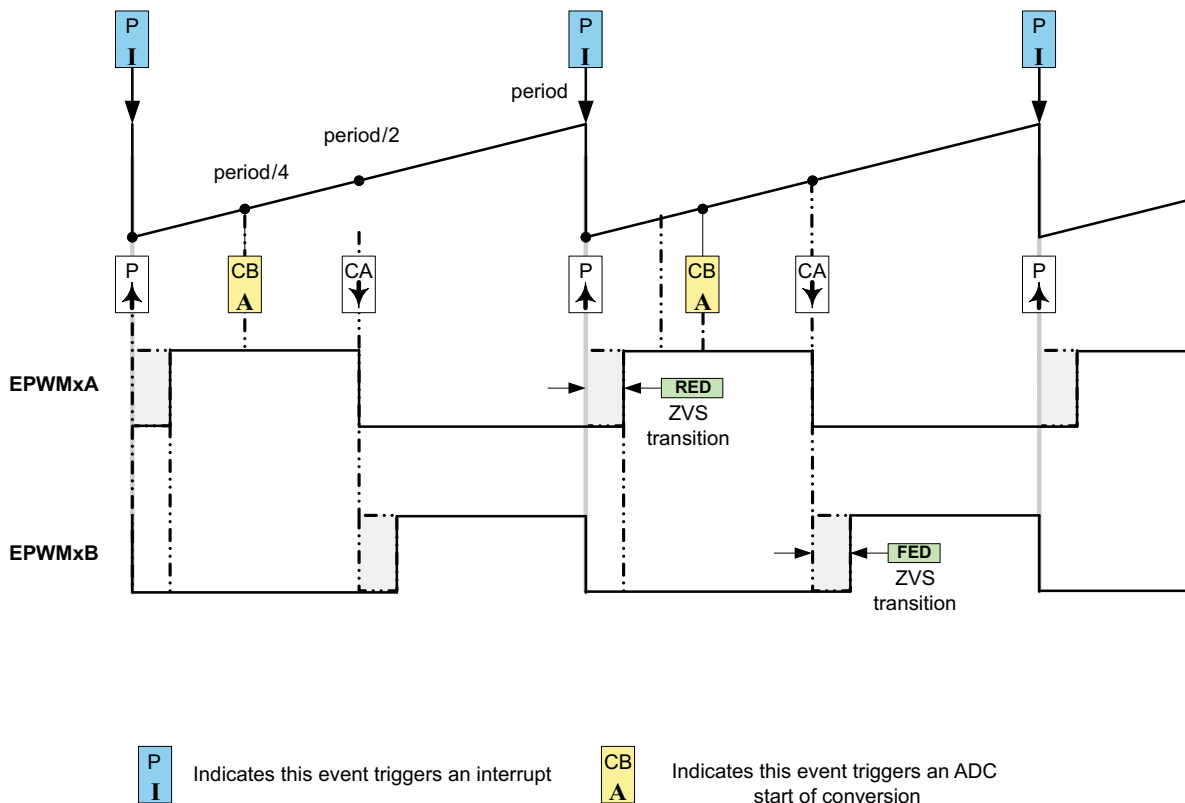
Various topologies of resonant converters are well-known in the field of power electronics for many years. In addition to these, H-bridge LLC resonant converter topology has recently gained popularity in many consumer electronics applications where high efficiency and power density are required. In this example single channel configuration of ePWM1 is detailed, yet the configuration can easily be extended to multi channel. Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead it is frequency. Although the deadband is not controlled and kept constant as 300ns (that is, 30 @ 100MHz TBCLK), it is up to user to update it in real time to enhance the efficiency by adjusting enough time delay for soft switching.

Figure 18-77. Control of Two Resonant Converter Stages



NOTE:  $\Theta = X$  indicates value in phase register is 'don't care'

Figure 18-78. H-Bridge LLC Resonant Converter PWM Waveforms



### 18.14 Register Lock Protection

The register lock protection mechanism is added to protect the critical ePWM registers from being corrupted by accidental writes in case of runaway code. The register EPWMLOCK contains the definition of Lock bits (Table 18-14 shows the lock bits and the corresponding registers). This register also has a KEY field; writes to this register will succeed only if the KEY field is written with a value of 0xa5a5. Refer to the register descriptions for more details.

**Table 18-14. Lock Bits and Corresponding Registers**

Bit Field	Definition	Registers Locked
HRLOCK	HRPWM Register Set Lock	HRCNFG, HRPWR, HRMSTEP, HRPCTL
GLLOCK	Global Load Register Set Lock	GLDCTL, GLDCFG
TZCFGLOCK	TripZone Register Set Lock	TZSEL, TZDCSEL, TZCTL, TZCTL2, TZCTLDCA, TZCTLDCB, TZEINT
TZCLRLOCK	TripZone Clear Register Set Lock	TZCLR, TZCBCCLR, TZOSTCLR, TZFRC
DCLOCK	Digital Compare Register Set Lock	DCTRIPSEL, DCACTL, DCBCTL, DCFCTL, DCCAPCTL, DCAHTRIPSEL, DCALTRIPSEL, DCBHTRIPSEL, DCBLTRIPSEL

---

**NOTE:** Due to the presence of the KEY field in the same register, only 32-bit writes will succeed if the KEY matches. The 16-bit writes to the upper or lower half of this register will be ignored.

---

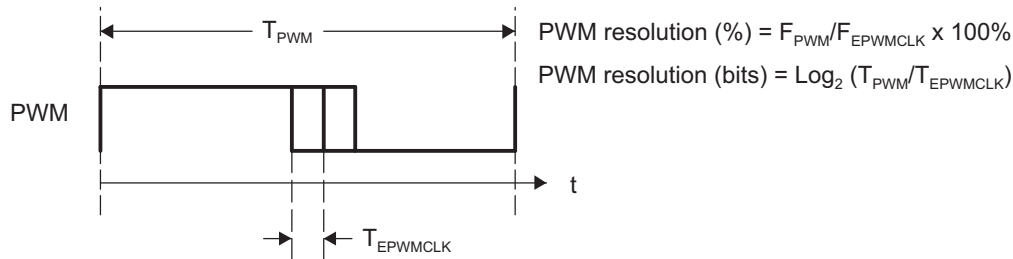
### 18.15 High-Resolution Pulse Width Modulator (HRPWM)

This module extends the time resolution capabilities of the conventionally derived digital pulse width modulator (PWM). HRPWM is typically used when PWM resolution falls below ~ 9-10 bits. The key features of HRPWM are:

- Extended time resolution capability
- Used in both duty cycle and phase-shift control methods
- Finer time granularity control or edge positioning using extensions to the Compare A, Compare B and Phase registers
- Implemented using the A & B signal path of PWM, that is, on the EPWMxA and EPWMxB output.
- Dead band high-resolution control for falling and rising edge delay in half cycle clocking operation
- Self-check diagnostics software mode to check if the micro edge positioner (MEP) logic is running optimally
- Enables high resolution output swapping on the EPWMxA and EPWMxB output
- Enables high-resolution output on EPWMxB signal output via inversion of EPWMxA signal output
- Enables high-resolution period, duty and phase control on the EPWMxA and EPWMxB output on devices with an ePWM module. See the device-specific data manual to determine if your device has an ePWM module for high-resolution period support.

The ePWM peripheral is used to perform a function mathematically equivalent to a digital-to-analog converter (DAC). As shown in [Figure 18-79](#), the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.

**Figure 18-79. Resolution Calculations for Conventionally Generated PWM**



If the required PWM operating frequency does not offer sufficient resolution in PWM mode, you may want to consider HRPWM. As an example of improved performance offered by HRPWM, the table below shows resolution in bits for various PWM frequencies. These values assume a MEP step size of 180 ps. See the device-specific data sheet for typical and maximum performance specifications for the MEP.

**Table 18-15. Resolution for PWM and HRPWM**

PWM Freq (kHz)	Regular Resolution (PWM)		High Resolution (HRPWM)	
	100 MHz EPWMCLK		Bits	%
20	12.3	0.02	18.1	0.000
50	11	0.05	16.8	0.001
100	10	0.1	15.8	0.002
150	9.4	0.15	15.2	0.003
200	9	0.2	14.8	0.004
250	8.6	0.25	14.4	0.005
500	7.6	0.5	13.4	0.009
1000	6.6	1	12.4	0.018
1500	6.1	1.5	11.9	0.027
2000	5.6	2	11.4	0.036

Although each application may differ, typical low frequency PWM operation (below 250 kHz) may not require HRPWM. HRPWM capability is most useful for high frequency PWM requirements of power conversion topologies such as:

- Single-phase buck, boost, and flyback
- Multi-phase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

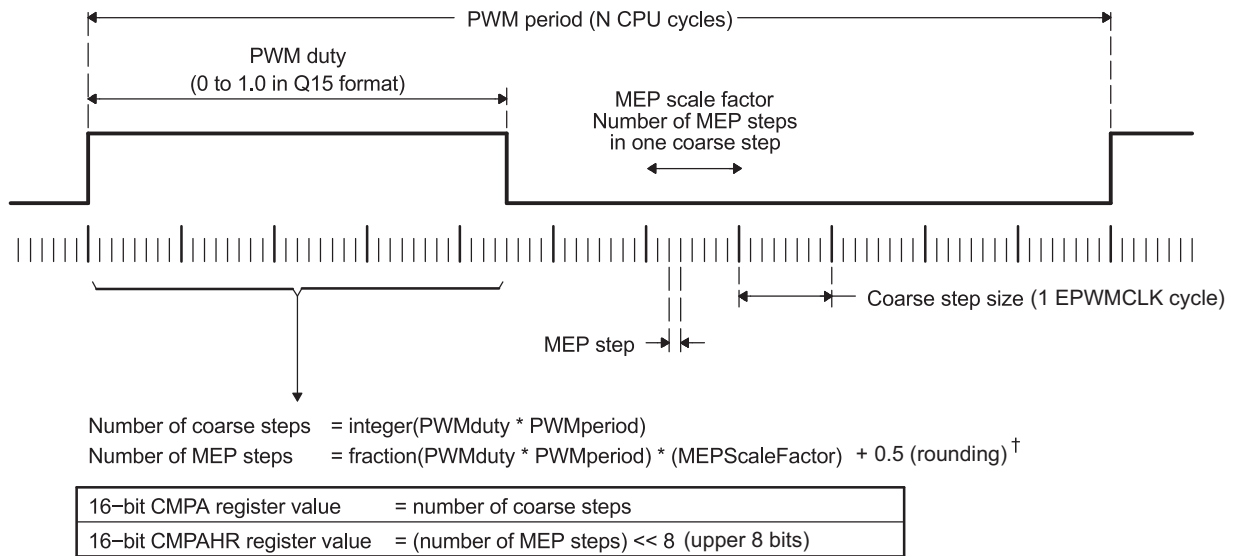


### 18.15.1 Operational Description of HRPWM

The HRPWM is based on micro-edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of a conventional PWM generator. The time step accuracy is on the order of 150 ps. See the device-specific data sheet for the typical MEP step size on a particular device. The HRPWM also has a self-check software diagnostics mode to check if the MEP logic is running optimally, under all operating conditions. Details on software diagnostics and functions are in [Section 18.15.1.7](#).

The figure below shows the relationship between one coarse system clock and edge position in terms of MEP steps, which are controlled via an 8-bit field in the Compare A extension register (CMPAHR). The same operating logic applies to CMPBHR as well.

**Figure 18-80. Operating Logic Using MEP**



† For MEP range and rounding adjustment. (0x0080 in Q8 format)

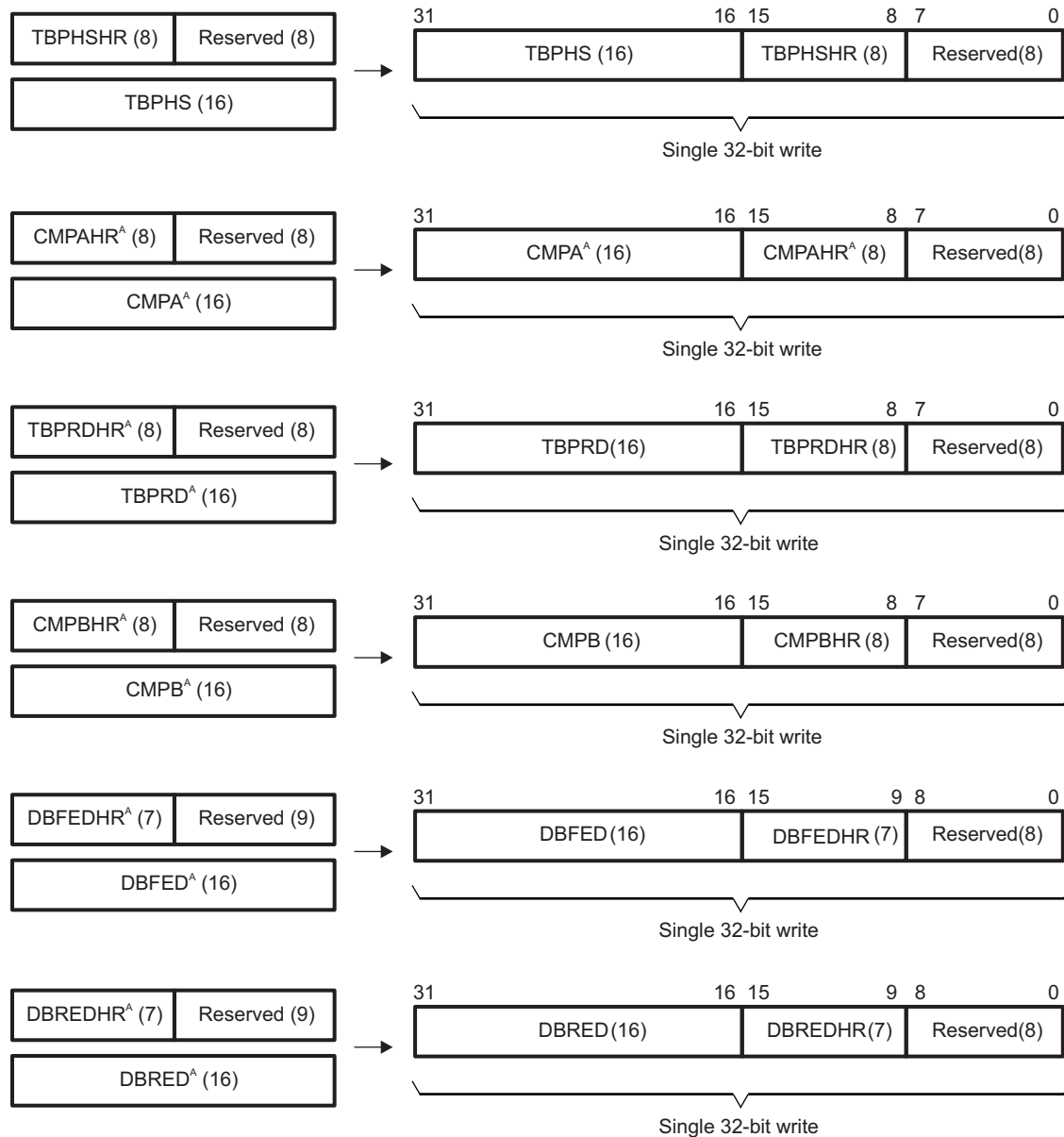
To generate an HRPWM waveform, configure the ePWM registers as you would to generate a conventional PWM of a given frequency and polarity. The HRPWM works together with the ePWM registers to extend edge resolution. Although many programming combinations are possible, only a few are needed and practical. These methods are described in [Section 18.15.1.8](#).

#### 18.15.1.1 Controlling the HRPWM Capabilities

The MEP of the HRPWM is controlled by six extension registers. These HRPWM registers are concatenated with the 16-bit TBPHS, TBPRD, CMPA, CMPBM, DBREDM & DBFEDM registers used to control PWM operation.

- TBPHSHR - Time Base Phase High Resolution Register
- CMPAHR - Counter Compare A High Resolution Register; CMPAHR is for use with the AQ output of Channel A, and is not related to CMPA
- TBPRDHR - Time Base Period High Resolution Register. (available on some devices)
- CMPBHR - Counter Compare B High Resolution Register; CMPBHR is for use with the AQ output of Channel B, and is not related to CMPB
- DBREDHR - Deadband Generator Rising Edge Delay High Resolution Register
- DBFEDHR - Deadband Generator Falling Edge Delay High Resolution Register

**NOTE:** TBPHSHR must not be used. Instead TRREM (translator remainder register) must be used to mimic the functionality of TBPHSHR.

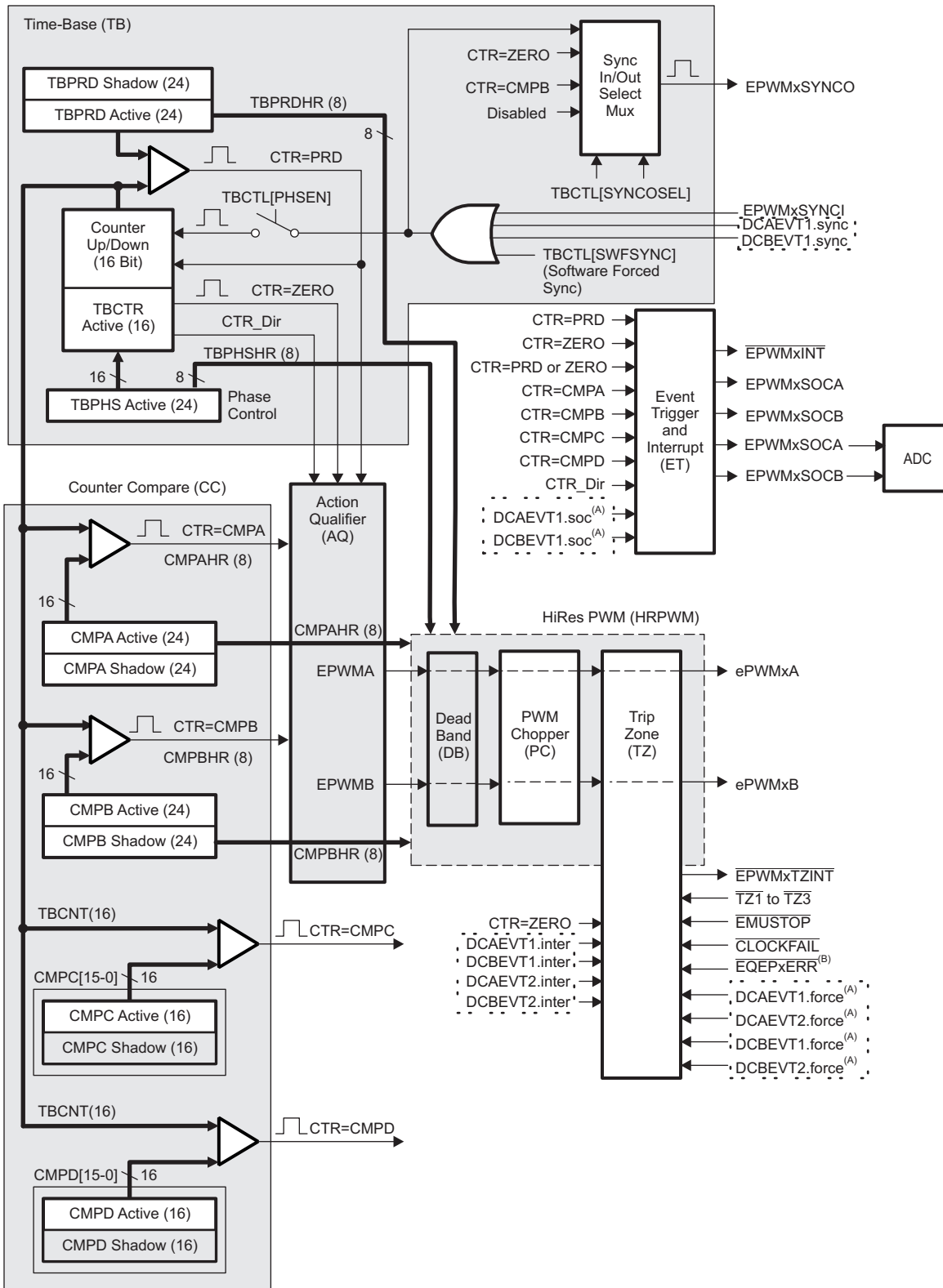
**Figure 18-81. HRPWM Extension Registers and Memory Configuration**


- A Dependant upon your device, these registers may be mirrored and can be written to at two different memory locations. Check the device-specific Technical Reference Manual's *ePWM* chapter for more details on how to read and write to these locations.

**NOTE:** HRPWM capabilities on Deadband Rising Edge Delay and Falling Edge Delay is applicable only during Dead Band half cycle clocking Operation. The number of MEP steps is half in size [ bits 15:9 ]than duty and phase high resolution registers for the same reason

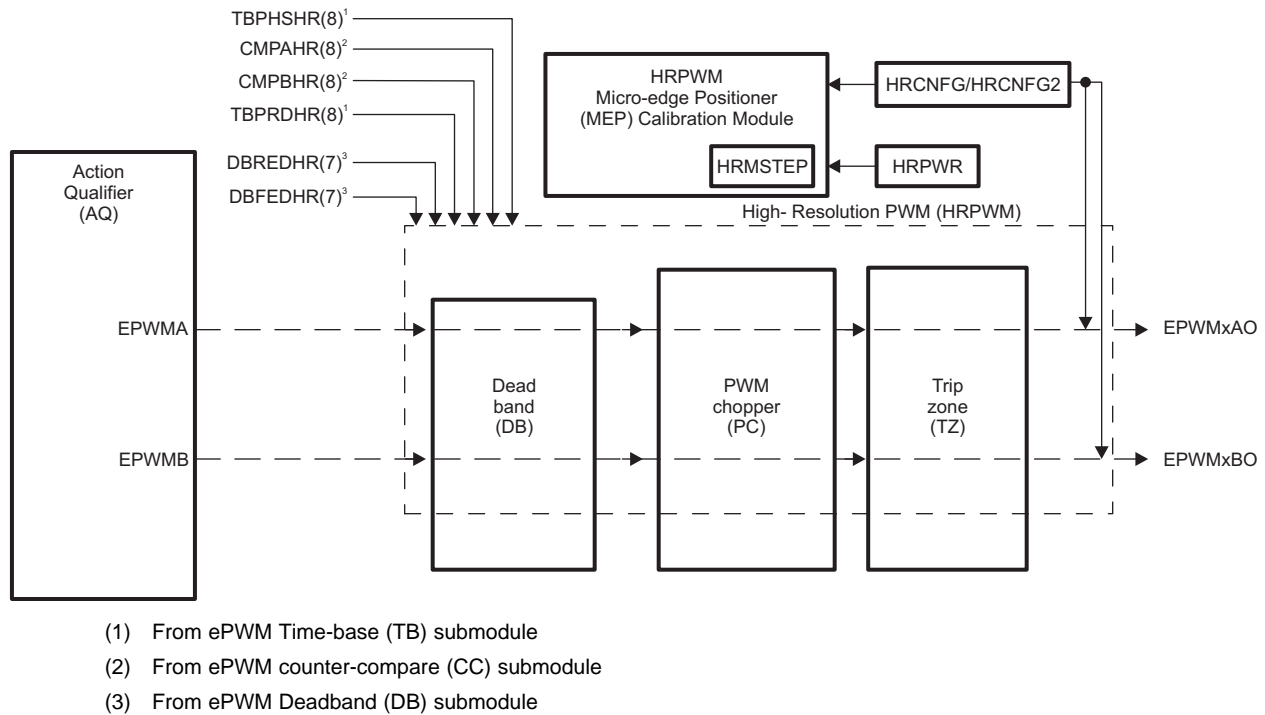
HRPWM capabilities are controlled using the Channel A & B PWM signal path. HRPWM support on the Dead band signal path is available by properly configuring the HRCNFG2 register. The following figure shows how the HRPWM interfaces with the 8-bit extension registers.

Figure 18-82. HRPWM System Interface



A These events are generated by the ePWM digital compare (DC) submodule.

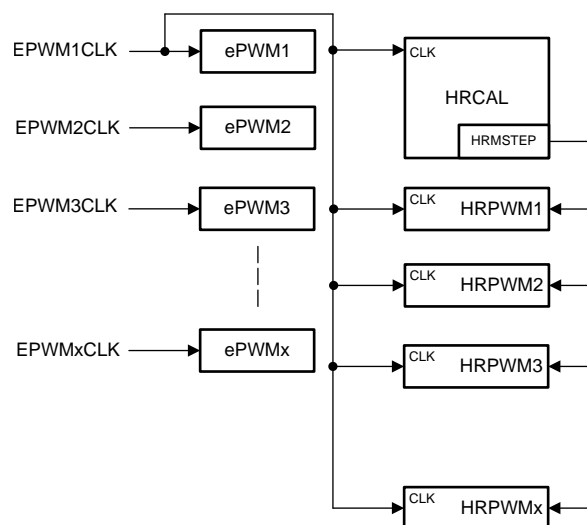
Figure 18-83. HRPWM Block Diagram



### 18.15.1.2 HRPWM Source Clock

The HRPWM module and HRCAL module are clocked from the EPWM1CLK. Therefore EPWM1CLK must be enabled for the HRCAL or any of the HRPWM modules to be enabled. The figure Figure 18-84 shows the HRCAL and HRPWM modules are sourced from ePWM1 clock source.

Figure 18-84. HRPWM and HRCAL Source Clock



### 18.15.1.3 Configuring the HRPWM

Once the ePWM has been configured to provide conventional PWM of a given frequency and polarity, the HRPWM is configured by programming the HRCNFG register in that particular ePWM module's register space. This register provides the following configuration options:

**Edge Mode** — The MEP can be programmed to provide precise position control on the rising edge (RE), falling edge (FE) or both edges (BE) at the same time. FE and RE are used for power topologies requiring duty cycle control (CMPA or CMPB high-resolution control), while BE is used for topologies requiring phase shifting, for example, phase shifted full bridge (TBPHS or TBPRD high-resolution control).

**Control Mode** — The MEP is programmed to be controlled either from the CMPAHR / CMPBHR register in case of duty cycle control or the TBPHSHR register (phase control). RE or FE control mode should be used with CMPAHR or CMPBHR register. BE control mode should be used with TBPHSHR register. When the MEP is controlled from the TBPRDHR register (period control) the duty cycle and phase can also be controlled via their respective high-resolution registers.

**Shadow Mode** — This mode provides the same shadowing (double buffering) option as in regular PWM mode. This option is valid only when operating from the CMPAHR, CMPBHR and TBPRDHR registers and should be chosen to be the same as the regular load option for the CMPA, CMPB register. If TBPHSHR is used, then this option has no effect.

**High-Resolution B Signal Control** — The B signal path of an ePWM channel can generate a high-resolution output by outputting an inverted version of the high-resolution ePWMxA signal on the ePWMxB pin. A Type 2, or Type 4 HRPWM module can also enable high-resolution features on the B signal path independently of the A signal path as well.

**Swap ePWMxA and ePWMxB Outputs** — This mode enables the swapping of the high resolution A & B outputs. The mode selection allows either A & B Outputs Unchanged or A Output Comes Out On B and B Output Comes Out On A

**Auto-conversion Mode** — This mode is used in conjunction with the scale factor optimization (SFO) software only. For a type 4 HRPWM module, below is a description of the Auto-conversion Mode taking CMPAHR as an example. If auto-conversion is enabled,  $CMPAHR = \text{fraction}(PWMduty * PWMperiod) \ll 8$ . The scale factor optimization software will calculate the MEP scale factor in background code and automatically update the HRMSTEP register with the calculated number of MEP steps per coarse step. The MEP Calibration Module will then use the values in the HRMSTEP and CMPAHR register to automatically calculate the appropriate number of MEP steps represented by the fractional duty cycle and move the high-resolution ePWM signal edge accordingly. If auto-conversion is disabled, the CMPAHR register behaves like a type 0 HRPWM module and  $CMPAHR = (\text{fraction}(PWMduty * PWMperiod) * MEP \text{ Scale Factor} + 0.5) \ll 8$ . All calculations will need to be performed by user code in this mode, and the HRMSTEP register is ignored. Auto-conversion for high-resolution period has the same behavior as auto-conversion for high-resolution duty cycle. Auto-conversion must always be enabled for high-resolution period mode.

---

**NOTE:** Auto-conversion Mode performs the calculation for CMPBHR, DBREDHR and DBFEDHR as well. The scale factor optimization software will calculate the MEP scale factor in background code and automatically update the HRMSTEP register with the calculated number of MEP steps per coarse step. The MEP Calibration Module will then use the values in the HRMSTEP and CMPBHR or DBREDHR / DBFEDHR register to automatically calculate the appropriate number of MEP steps represented by the fractional components and move the high-resolution ePWM signal edge accordingly. If auto-conversion is disabled, CMPBHR behaves same as CMPAHR.  $CMPBHR = (\text{fraction}(PWMduty * PWMperiod) * MEP \text{ Scale Factor} + 0.5) \ll 8$ .

---

#### 18.15.1.4 Configuring Hi-Res in Deadband Rising Edge and Falling Edge Delay

Once the ePWM has been configured to provide conventional PWM of a given frequency, polarity and deadband enabled in half cycle clocking mode, the high resolution operation on dead band RED and FED lines are enabled by programming the HRCNFG2 register in that particular ePWM module's register space. This register provides the following configuration options:

**Edge Mode** — The MEP can be programmed to provide precise position control on the dead band rising edge (RED), dead band falling edge (FED) or both edges (rising edge of DBRED signal and falling edge of DBFED signal) at the same time.

**Control Mode** — Selects the time event that loads the shadow value in active register for DBRED and DBFED in high resolution mode. The user needs to select the pulse to match the selection in the ePWM DBCTL[LOADREDMODE] & DBCTL[LOADFEDMODE] bits .

#### 18.15.1.5 Principle of Operation

The MEP logic is capable of placing an edge in one of 255 (8 bits) discrete time steps (see device-specific data sheet for typical MEP step size). The MEP works with the TBM and CCM registers to be certain that time steps are optimally applied and that edge placement accuracy is maintained over a wide range of PWM frequencies, system clock frequencies and other operating conditions. [Table 18-16](#) shows the typical range of operating frequencies supported by the HRPWM.

**Table 18-16. Relationship Between MEP Steps, PWM Frequency and Resolution**

System (MHz)	MEP Steps Per EPWMCLK <sup>(1)(2)(3)</sup>	PWM MIN (Hz) <sup>(4)</sup>	PWM MAX (MHz)	Res. @ MAX (Bits) <sup>(5)</sup>
60.0	93	916	3.00	10.9
70.0	79	1068	3.50	10.6
80.0	69	1221	4.00	10.4
90.0	62	1373	4.50	10.3
100.0	56	1526	5.00	10.1

- (1) TBCLK = EPWMCLK.
- (2) Table data based on a MEP time resolution of 180 ps (this is an example value. See the device-specific data sheet for MEP limits)
- (3) MEP steps applied =  $T_{EPWMCLK}/180$  ps in this example.
- (4) PWM minimum frequency is based on a maximum period value, (TBPRD = 65535). PWM mode is asymmetrical up-count.
- (5) Resolution in bits is given for the maximum PWM frequency stated.

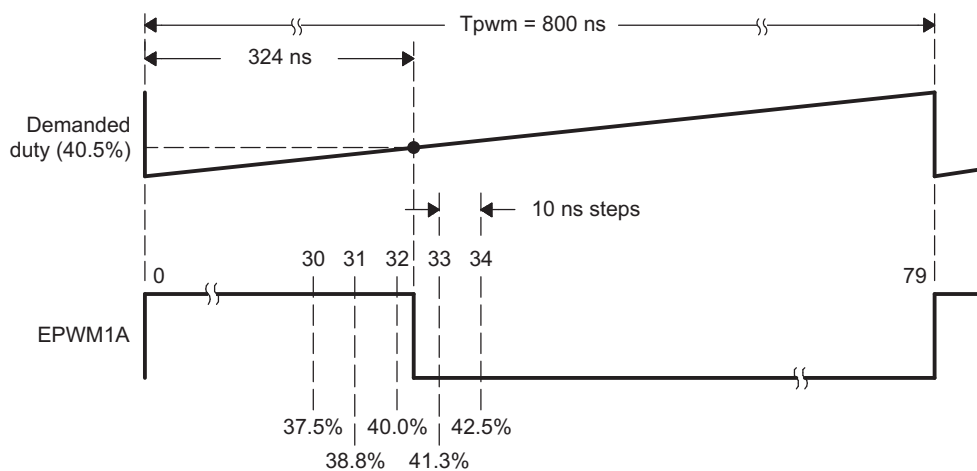
**18.15.1.5.1 Edge Positioning**

**NOTE:** The below example is presented using [CMPA:CMPAHR] register combination. The theory of operation and equations is same if the user intends to use [CMPBM:CMPBHRM] for duty cycle control.

In a typical power control loop, a digital controller issues a duty command, usually expressed in a per unit or percentage terms. Assume that for a particular operating point, the demanded duty cycle is 0.405 or 40.5% on time and the required converter PWM frequency is 1.25 MHz. In conventional PWM generation with a system clock of 100 MHz, the duty cycle choices are in the vicinity of 40.5%. As shown in Figure 18-85, a compare value of 32 counts (duty = 40%) is the closest to 40.5% that can be attained. This is equivalent to an edge position of 320 ns instead of the desired 324 ns. This data is shown in Table 18-17.

By utilizing the MEP, you can achieve an edge position much closer to the desired point of 324 ns. Table 18-17 shows that in addition to the CMPA value, 22 steps of the MEP (CMPAHR register) will position the edge at 323.96 ns, resulting in almost zero error. In this example, it is assumed that the MEP has a step resolution of 180 ps.

**Figure 18-85. Required PWM Waveform for a Requested Duty = 40.5%**



**Table 18-17. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right)**

CMPA (count) <sup>(1) (2) (3)</sup>	DUTY %	High Time (ns)	CMPA (count)	CMPAHR (count)	Duty (%)	High Time (ns)
28	<b>35.0</b>	280	32	18	<b>40.405</b>	323.24
29	<b>36.3</b>	290	32	19	<b>40.428</b>	323.42
30	<b>37.5</b>	300	32	20	<b>40.450</b>	323.60
31	<b>38.8</b>	310	32	21	<b>40.473</b>	323.78
32	<b>40.0</b>	320	32	22	<b>40.495</b>	323.96
33	<b>41.3</b>	330	32	23	<b>40.518</b>	324.14
34	<b>42.5</b>	340	32	24	<b>40.540</b>	324.32
			32	25	<b>40.563</b>	324.50
Required			32	26	<b>40.585</b>	324.68
32.40	<b>40.5</b>	324	32	27	<b>40.608</b>	324.86

<sup>(1)</sup> TBCLK = 100 MHz, 10 ns

<sup>(2)</sup> For a PWM Period register value of 80 counts, PWM Period = 80 x 10 ns = 800 ns, PWM frequency = 1/800 ns = 1.25 MHz

<sup>(3)</sup> Assumed MEP step size for the above example = 180 ps  
See the device-specific data manual for typical and maximum MEP values.

### 18.15.1.5.2 Scaling Considerations

The mechanics of how to position an edge precisely in time has been demonstrated using the resources of the standard CMPA and MEP (CMPAHR) registers. In a practical application, however, it is necessary to seamlessly provide the CPU a mapping function from a per-unit (fractional) duty cycle to a final integer (non-fractional) representation that is written to the [CMPA:CMPAHR] register combination.

To do this, first examine the scaling or mapping steps involved. It is common in control software to express duty cycle in a per-unit or percentage basis. This has the advantage of performing all needed math calculations without concern for the final absolute duty cycle, expressed in clock counts or high time in ns. Furthermore, it makes the code more transportable across multiple converter types running different PWM frequencies.

To implement the mapping scheme, a two-step scaling procedure is required.



**Assumptions for this example:**

TBCLK	=	10 ns (100 MHz)
PWM frequency	=	1.25 MHz (1/800 ns)
Required PWM duty cycle, <b>PWMDuty</b>	=	0.405 (40.5%)
PWM period in terms of coarse steps, <b>PWMPeriod</b> (800 ns/10 ns)	=	80
Number of MEP steps per coarse step at 180 ps (10 n/180 ps ), <b>MEP_ScaleFactor</b>	=	55
Value to keep CMPAHR within the range of 1-255 and fractional rounding constant (default value)	=	0.5 (0080h in Q8 format)

**Step 1: Percentage Integer Duty value conversion for CMPA register**

CMPA register value	=	$\text{int}(\text{PWMDuty} * \text{PWMPeriod})$ ; int means integer part
	=	$\text{int}(0.405 * 80)$
	=	$\text{int}(32.4)$
CMPA register value	=	32 (20h)

**Step 2: Fractional value conversion for CMPAHR register**

CMPAHR	=	$(\text{frac}(\text{PWMDuty} * \text{PWMPeriod}) * \text{MEP\_ScaleFactor} + 0.5) \ll 8$ ; frac means fractional part
	=	$(\text{frac}(32.4) * 55 + 0.5) \ll 8$ ; Shifting is to move the value to the high byte of CMPAHR.
	=	$(0.4 * 55 + 0.5) \ll 8$
	=	$(22 + 0.5) \ll 8$
	=	$22.5 * 256$ ; Shifting left by 8 is the same as multiplying by 256.
	=	5760 (1680h)
CMPAHR	=	1680h CMPAHR value = 1600h (lower 8 bits will be ignored by hardware).

---

**NOTE:** If the AUTOCONV bit (HRCNFG.6) is set and the MEP\_ScaleFactor is in the HRMSTEP register, then CMPAHR / CMPBHR register value =  $\text{frac}(\text{PWMDuty} * \text{PWMperiod} \ll 8)$ . The rest of the conversion calculations are performed automatically in hardware, and the correct MEP-scaled signal edge appears on the ePWM channel output. If AUTOCONV is not set, the above calculations must be performed by software.

---

**NOTE:** The MEP scale factor (MEP\_ScaleFactor) varies with the system clock and DSP operating conditions. TI provides an MEP scale factor optimizing (SFO) software C function, which uses the built-in diagnostics in each HRPWM and returns the best scale factor for a given operating point.

The scale factor varies slowly over a limited range so the optimizing C function can be run very slowly in a background loop.

The CMPA, CMPB, CMPAHR and CMPBHR registers are configured in memory so that the 32-bit data capability of the 28x CPU can write this as a single concatenated value, that is, [CMPA:CMPAHR], [CMPB:CMPBHR], and so on.

The mapping scheme has been implemented in both C and assembly, as shown in [Section 18.15.1.8](#). The actual implementation takes advantage of the 32-bit CPU architecture of the 28xx, and is somewhat different from the steps shown in [Section 18.15.1.5.2](#).

For time-critical control loops where every cycle counts, the assembly version is recommended. This is a cycle optimized function (11 EPWMCLK cycles) that takes a Q15 duty value as input and writes a single [CMPA:CMPAHR] value.

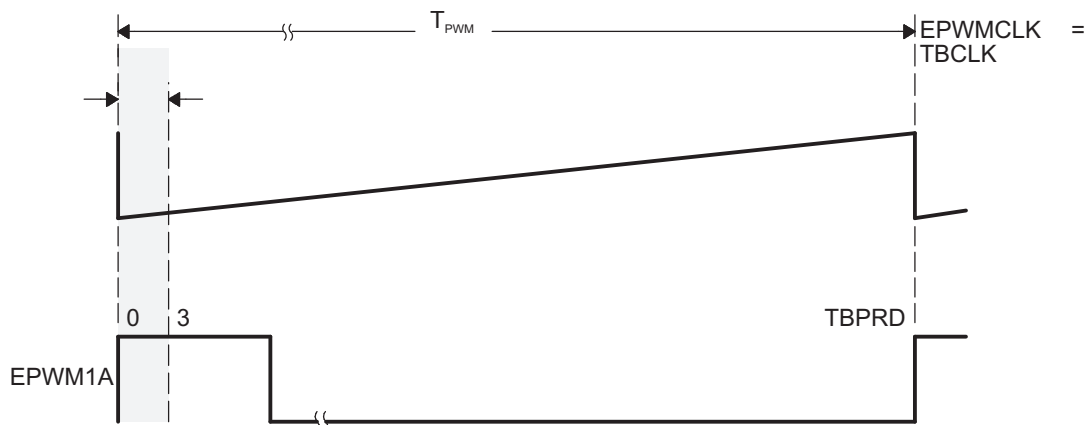
### 18.15.1.5.3 Duty Cycle Range Limitation

In high resolution mode, the MEP is not active for 100% of the PWM period. It becomes operational:

- Three EPWMCLK cycles after the period starts when high-resolution period (TBPRDHR) control is not enabled.
- When high resolution period (TBPRDHR) control is enabled via the HRPCTL register:
  - In up-count mode: three EPWMCLK cycles after the period starts until three EPWMCLK cycles before the period ends.
  - In up-down count mode: when counting up, three cycles after CTR = 0 until three cycles before CTR = PRD, and when counting down, three cycles after CTR = PRD until three cycles before CTR = 0.
- When using DBREDHR or DBFEDHR, DBRED and/or DBFED (the register corresponding to the edge with hi-resolution displacement) must be greater than or equal to 7.

Duty cycle range limitations are illustrated in [Figure 18-86](#) to [Figure 18-89](#). This limitation imposes a duty cycle limit on the MEP. For example, precision edge control is not available all the way down to 0% duty cycle. When high-resolution period control is disabled, regular PWM duty control is fully operational down to 0% duty cycle despite the unavailability of HRPWM features in the first three cycles. In most applications this should not be an issue as the controller regulation point is usually not designed to be close to 0% duty cycle. To better understand the useable duty cycle range, see [Table 18-18](#). When high-resolution period control is enabled (HRPCTL[HRPE]=1), the duty cycle must not fall within the restricted range. Otherwise, there may be undefined behavior on the ePWMxA output.

**Figure 18-86. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)**



**Table 18-18. Duty Cycle Range Limitation for Three EPWMCLK/TBCLK Cycles**

PWM Frequency <sup>(1)</sup> (kHz)	3 Cycles Minimum Duty	3 Cycles Maximum Duty <sup>(2)</sup>
200	0.6%	99.4%
400	1.2%	98.8%
600	1.8%	98.2%
800	2.4%	97.6%
1000	3%	97%
1200	3.6%	96.4%
1400	4.2%	95.8%
1600	4.8%	95.2%
1800	5.4%	94.6%
2000	6%	94%

<sup>(1)</sup> EPWMCLK = TBCLK = 100 MHz

<sup>(2)</sup> This limitation applies only if high-resolution period (TBPRDHR) control is enabled.

If the application demands HRPWM operation below the minimum duty cycle limitation, then the HRPWM can be configured to operate in count-down mode with the rising edge position (REP) controlled by the MEP when high-resolution period is disabled (HRPCTL[HRPE] = 0). This is illustrated in [Figure 18-87](#). In this configuration, the minimum duty cycle limitation is no longer an issue. However, there will be a maximum duty limitation with same percent numbers as given in [Table 18-18](#).

Figure 18-87. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)

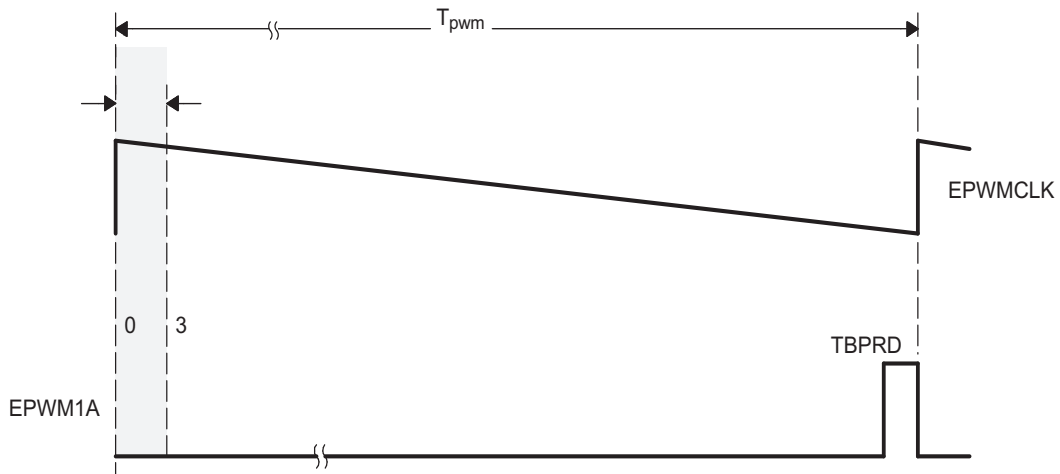


Figure 18-88. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)

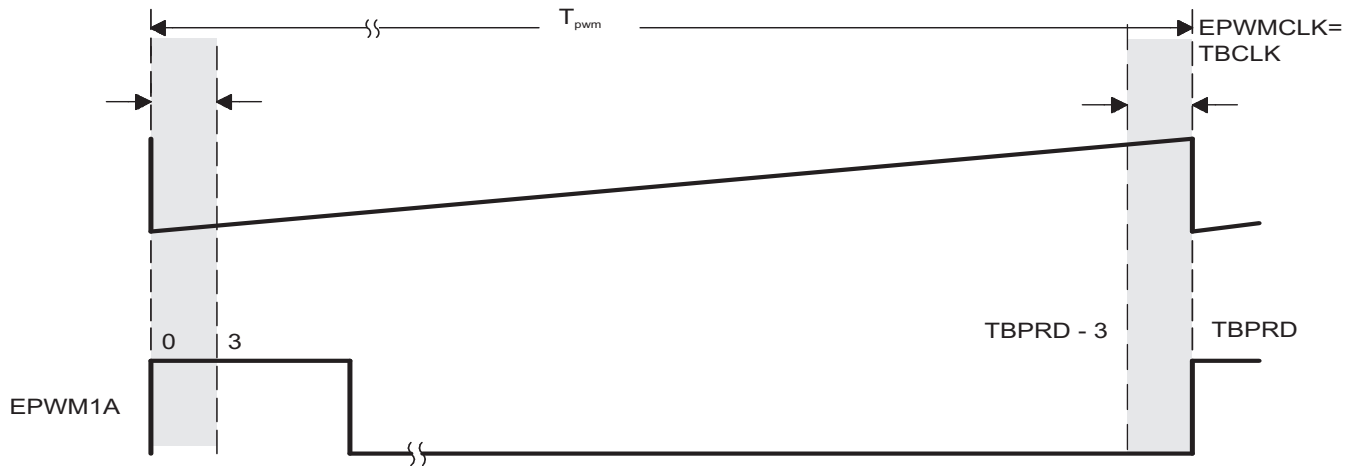
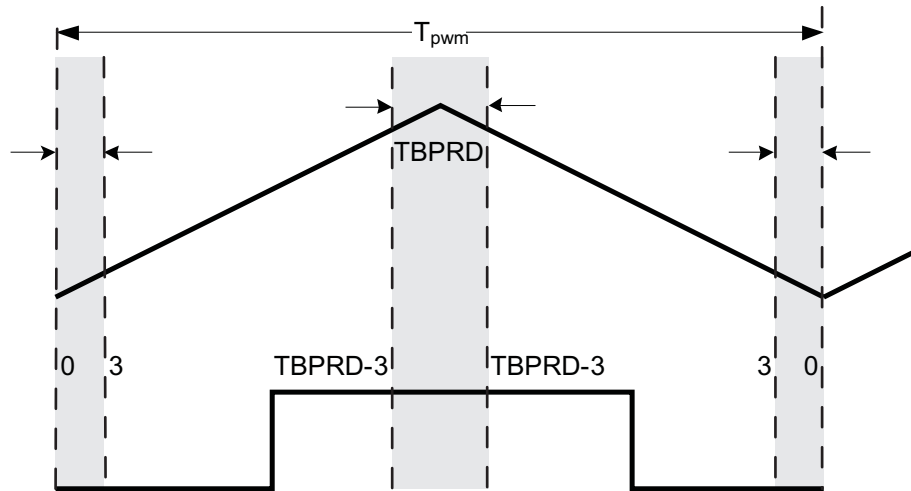


Figure 18-89. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)



**NOTE:** If the application has enabled high-resolution period control (HRPCTL[HRPE]=1), the duty cycle must not fall within the restricted range. Otherwise, there will be undefined behavior on the ePWM output.

**18.15.1.5.4 High Resolution Period**

High resolution period control using the MEP logic is supported on devices with a Type 1 ePWM module or greater.

---

**NOTE:** When high-resolution period control is enabled, on ePWMxA only, and not ePWMxB output and vice versa, the non hi-res output will have +/- 1 TBCLK cycle jitter in up-count mode and +/- 2 TBCLK cycle jitter in up-down count mode.

---

The scaling procedure described for duty cycle in [Section 18.15.1.5.2](#) applies for high-resolution period as well:

**Assumptions for this example:**

TBCLK	= 10 ns (100 MHz)
Required PWM frequency	= 175 kHz (period of 571.428)
Number of MEP steps per coarse step at 180 ps (MEP_ScaleFactor)	= 55 (10 ns / 180 ps)
Value to keep TBPRDHR within range of 1-255 and fractional rounding constant (default value)	= 0.5 (0080h in Q8 format)

**Problem:**

In up-count mode:

If TBPRD = 571, then PWM frequency = 174.82 kHz (period =  $(571+1) * T_{TBCLK}$ ).

If TBPRD = 570, then PWM frequency = 175.13 kHz (period =  $(570+1) * T_{TBCLK}$ ).

In up-down count mode:

If TBPRD = 286, then PWM frequency = 174.82 kHz (period =  $(286*2) * T_{TBCLK}$ ).

If TBPRD = 285, then PWM frequency = 175.44 kHz (period =  $(285*2) * T_{TBCLK}$ ).

**Solution:**

With 55 MEP steps per coarse step at 180 ps each:

**Step 1: Percentage Integer Period value conversion for TBPRD register**

$$\begin{aligned} \text{Integer period value} &= 571 * T_{TBCLK} \\ &= \text{int}(571.428) * T_{TBCLK} \\ &= \text{int}(\text{PWMperiod}) * T_{TBCLK} \end{aligned}$$

In up-count mode:

$$\begin{aligned} \text{TBPRD} &= 570 \text{ (TBPRD = period value - 1)} \\ &= 023Ah \end{aligned}$$

In up-down count mode:

$$\begin{aligned} \text{TBPRD} &= 285 \text{ (TBPRD = period value / 2)} \\ &= 011Dh \end{aligned}$$

**Step 2: Fractional value conversion for TBPRDHR register**

$$\text{TBPRDHR register value} = (\text{frac}(\text{PWMperiod}) * \text{MEP\_ScaleFactor} + 0.5)$$

If auto-conversion enabled and HRMSTEP =

$$\text{MEP\_ScaleFactor value (55):} = \text{frac}(\text{PWMperiod}) \ll 8 \text{ (Shifting is to move the value to the high byte of TBPRDHR)}$$

$$\begin{aligned} \text{TBPRDHR register value} &= \text{frac}(571.428) \ll 8 \\ &= 0.428 \times 256 \\ &= 6D00h \end{aligned}$$

The autoconversion will then automatically perform the calculation such that TBPRDHR MEP delay is scaled by hardware to:

$$\begin{aligned} &= ((\text{TBPRDHR}(15:0) \gg 8) \times \text{HRMSTEP} + 80\text{h}) \ll 8 \\ &= (006\text{Dh} \times 55 + 80\text{h}) \gg 8 \\ &= (17\text{EBh}) \gg 8 \\ \text{Period MEP delay} &= 0017\text{h MEP Steps} \end{aligned}$$

#### 18.15.1.5.4.1 High-Resolution Period Configuration

To use High Resolution Period, the ePWMx module must be initialized in the exact order presented.

The steps below use CMPA with shadow registers and the corresponding HRCNFG bits for hi-resolution operation on EPWMxA. For hi-resolution operation on EPWMxB, make the appropriate substitutions with the B channel fields.

1. Enable ePWMx clock
2. Enable HRPWM clock
3. Disable TBCLKSYNC
4. Configure ePWMx registers - AQ, TBPRD, CC, and so on.
  - ePWMx may only be configured for up-count or up-down count modes. High-resolution period is not compatible with down-count mode.
  - TBPRD and CC registers must be configured for shadow loads.
  - CMPCTL[LOADAMODE]
    - In up-count mode: CMPCTL[LOADAMODE] = 1 (load on CTR = PRD)
    - In up-down count mode: CMPCTL[LOADAMODE] = 2 (load on CTR=0 or CTR=PRD)
5. Configure the HRCNFG register such that:
  - HRCNFG[HRLOAD] = 2 (load on either CTR = 0 or CTR = PRD)
  - HRCNFG[AUTOCONV] = 1 (Enable auto-conversion)
  - HRCNFG[EDGMODE] = 3 (MEP control on both edges)
6. For TBPHS:TBPHSHR synchronization with high-resolution period, set both HRPCTL[TBPSHRLOADE] = 1 and TBCTL[PHSEN] = 1. In up-down count mode these bits must be set to 1 regardless of the contents of TBPHSHR.
7. Enable high-resolution period control (HRPCTL[HRPE] = 1)
8. Enable TBCLKSYNC
9. TBCTL[SWFSYNC] = 1
10. HRMSTEP must contain an accurate MEP scale factor (# of MEP steps per EPWMCLK coarse step) because auto-conversion is enabled. The MEP scale factor can be acquired via the SFO() function described in [Section 18.15.2](#).
11. To control high-resolution period, write to the TBPRDHR(M) registers.

---

**NOTE:** When high-resolution period mode is enabled, an EPWMxSYNC pulse will introduce +/- 1 - 2 cycle jitter to the PWM (+/- 1 cycle in up-count mode and +/- 2 cycle in up-down count mode). For this reason, TBCTL[SYNCOSEL] should not be set to 1 (CTR = 0 is EPWMxSYNCO source) or 2 (CTR = CMPB is EPWMxSYNCO source). Otherwise, the jitter will occur on every PWM cycle with the synchronization pulse.

When TBCTL[SYNCOSEL] = 0 (EPWMxSYNCl is EPWMxSYNCO source), a software synchronization pulse should be issued only once during high-resolution period initialization. If a software sync pulse is applied while the PWM is running, the jitter will appear on the PWM output at the time of the sync pulse.

---

**18.15.1.6 Deadband High Resolution Operation**
**Assumptions for this example:**

System clock	= 10 ns (100 MHz) &
Deadband Enabled in half cycle mode, TBCLK = EPWMCLK	
Required PWM frequency	1.33MHz (1 / 750 ns)
Required PWM duty cycle	0.5 (50%)
Required Dead band Rising Edge Delay	5% over duty
Required Dead band Rising Edge Delay in ns	(0.05 * 375 ns ) = 18.75 ns

---

**NOTE:** Just like the duty cycle restrictions when using HRPWM, the DBRED and DBFED values must be greater than 3 to use hi-res deadband.

---

**Deadband Delay Values as a Function of DBFED and DBRED:**

When half-cycle clocking is enabled, the formula to calculate the falling-edge-delay and rising-edge-delay becomes:

$$FED = DBFED * TBCLK / 2$$

$$RED = DBRED * TBCLK / 2$$

**DBRED and DBFED Calculated Values:**

Required Dead band Rising Edge Delay in ns = 18.75 ns

$$DBRED = RED / (TBCLK / 2)$$

$$DBRED = 18.75 \text{ ns} / 5 \text{ ns}$$

$$DBRED \text{ Required} = 3.75 \text{ ns}$$

With 55 MEP steps per coarse step at 180 ps each:

**Step 1: Integer Dead band value conversion for DBREDM register**

Integer DBRED value	= int (RED / (TBCLK / 2))
	= int (3.75)
DBRED	= 3

**Step 2: Fractional value conversion for Dead band high resolution register DBREDHR**

DBREDHR register value	= (frac(DBRED Required) * MEP_ScaleFactor + 0.5) << 8 (Shifting is to move the value to the high byte of DBREDHR)
	= (frac (3.75) * 55 + 0.5) << 8
	= ( 0.75 * 55 + 0.5 ) << 8
	= (41.75) * 256 Shifting left by 8 is the same as multiplying by 256.
DBREDHR value	= 29C0h MEP Steps
	Hardware will ignore lower 9 bits in the above calculated DBREDHR value



---

**NOTE:** If the AUTOCONV bit (HRCNFG.6) is set and the MEP\_ScaleFactor is in the HRMSTEP register, then DBREDHR:DBRED =  $\text{frac}(\text{required DB value}) < <8$ . The rest of the conversion calculations are performed automatically in hardware, and the correct MEP-scaled signal edge appears on the ePWM channel output. If AUTOCONV is not set, the above calculations must be performed by software.

---

### 18.15.1.7 Scale Factor Optimizing Software (SFO)

The micro edge positioner (MEP) logic is capable of placing an edge in one of 255 discrete time steps. As previously mentioned, the size of these steps is on the order of 150 ps (see device-specific data sheet for typical MEP step size on your device). The MEP step size varies based on worst-case process parameters, operating temperature, and voltage. MEP step size increases with decreasing voltage and increasing temperature and decreases with increasing voltage and decreasing temperature. Applications that use the HRPWM feature should use the TI-supplied MEP scale factor optimization (SFO) software function. The SFO function helps to dynamically determine the number of MEP steps per EPWMCLK period while the HRPWM is in operation.

To utilize the MEP capabilities effectively, the correct value for the MEP scaling factor needs to be known by the software. To accomplish this, the HRPWM module has built in self-check and diagnostic capabilities that can be used to determine the optimum MEP scale factor value for any operating condition. TI provides a C-callable library containing one SFO function that utilizes this hardware and determines the optimum MEP scale factor. As such, MEP control and diagnostics registers are reserved for TI use.

A detailed description of the SFO library - SFO\_TI\_Build\_V8.lib software can be found in [Section 18.15.2](#).

### 18.15.1.8 HRPWM Examples Using Optimized Assembly Code.

The best way to understand how to use the HRPWM capabilities is through two real examples:

1. Simple buck converter using asymmetrical PWM (count-up) with active high polarity.
2. DAC function using simple R+C reconstruction filter.

The following examples all have initialization and configuration code written in C. To make these easier to understand, the #defines shown below are used. Note, #defines introduced in the device-specific *Pulse Width Modulator (ePWM) Module Reference Guide* are also used.

**Example 18-2** This example assumes MEP step size of 150 ps and does not use the SFO library.

**Example 18-2. #Defines for HRPWM Header Files**

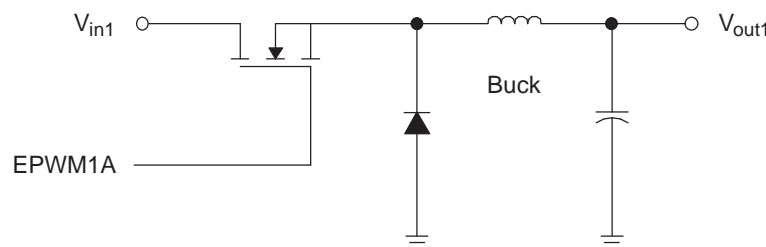
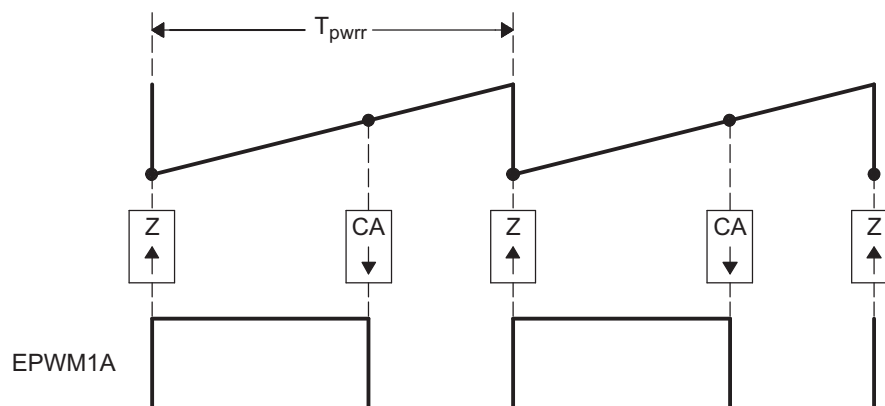
```
// HRPWM (High Resolution PWM) //
=====
// HRCNFG
#define HR_Disable 0x0
#define HR_REP 0x1 // Rising Edge position
#define HR_FEP 0x2 // Falling Edge position
#define HR_BEP 0x3 // Both Edge position #define HR_CMP 0x0 // CMPAHR controlled
#define HR_PHS 0x1 // TBPHSHR controlled #define HR_CTR_ZERO 0x0 // CTR = Zero event
#define HR_CTR_PRD 0x1 // CTR = Period event
#define HR_CTR_ZERO_PRD 0x2 // CTR = ZERO or Period event
#define HR_NORM_B 0x0 // Normal ePWMxB output
#define HR_INVERT_B 0x1 // ePWMxB is inverted ePWMxA output
```

**18.15.1.8.1 Implementing a Simple Buck Converter**

In this example, the PWM requirements are:

- PWM frequency = 1 MHz (that is, TBPRD = 100 )
- PWM mode = asymmetrical, up-count
- Resolution = 12.7 bits (with a MEP step size of 150 ps)

Figure 18-90 and Figure 18-91 show the required PWM waveform. As explained previously, configuration for the ePWM1 module is almost identical to the normal case except that the appropriate MEP options need to be enabled/selected.

**Figure 18-90. Simple Buck Controlled Converter Using a Single PWM**

**Figure 18-91. PWM Waveform Generated for Simple Buck Controlled Converter**


The example code shown consists of two main parts:

- Initialization code (executed once)
- Run time code (typically executed within an ISR)

[Example 18-3](#) shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

This example assumes MEP step size of 150 ps and does not use the SFO library.

### **Example 18-3. HRPWM Buck Converter Initialization Code**

```
void HrBuckDrvCnf(void)
{
// Config for conventional PWM first
EPwm1Regs.TBCTL.bit.PRDL = TB_IMMEDIATE;           // set Immediate load
EPwm1Regs.TBPRD = 100;                             // Period set for 1000 kHz PWM
hrbuck_period = 200;                               // Used for Q15 to Q0 scaling
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;           // EPWM1 is the Master
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
// Note: ChB is initialized here only for comparison purposes, it is not required

EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;     // optional
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;     // optional

EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;               // optional
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;             // optional
// Now configure the HRPWM resources
EALLOW;                                           // Note these registers are protected
                                                // and act only on ChA
EPwm1Regs.HRCNFG.all = 0x0;                       // clear all bits first
EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;           // Control Falling Edge Position
EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;           // CMPAHR controls the MEP
EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;       // Shadow load on CTR=Zero
EDIS;
MEP_ScaleFactor = 66*256;                         // Start with typical Scale Factor
                                                // value for 100 MHz
                                                // Note: Use SFO functions to update
                                                // MEP_ScaleFactor dynamically
}
```

[Example 18-4](#) shows an assembly example of run-time code for the HRPWM buck converter.

**Example 18-4. HRPWM Buck Converter Run-Time Code**

```

EPWM1_BASE .set 0x6800
CMPAHR1 .set EPWM1_BASE+0x8
;=====
HRBUCK_DRV; (can execute within an ISR or loop)
;=====
    MOVW DP, #_HRBUCK_In
    MOVL XAR2,@_HRBUCK_In      ; Pointer to Input Q15 Duty (XAR2)
    MOVL XAR3,#CMPAHR1        ; Pointer to HRPWM CMPA reg (XAR3)
; Output for EPWM1A (HRPWM)
    MOV T,*XAR2 ; T <= Duty
    MPYU ACC,T,@_hrbuck_period ; Q15 to Q0 scaling based on Period
    MOV T,@_MEP_ScaleFactor    ; MEP scale factor (from optimizer s/w)
    MPYU P,T,@AL               ; P <= T * AL, Optimizer scaling
    MOVH @AL,P                 ; AL <= P, move result back to ACC
    ADD ACC, #0x080            ; MEP range and rounding adjustment
    MOVL *XAR3,ACC             ; CMPA: CMPAHR(31:8) <= ACC
; Output for EPWM1B (Regular Res) Optional - for comparison purpose only
    MOV *+XAR3[2],AH          ; Store ACCH to regular CMPB

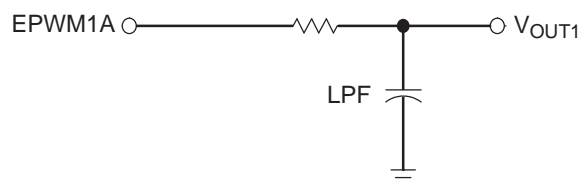
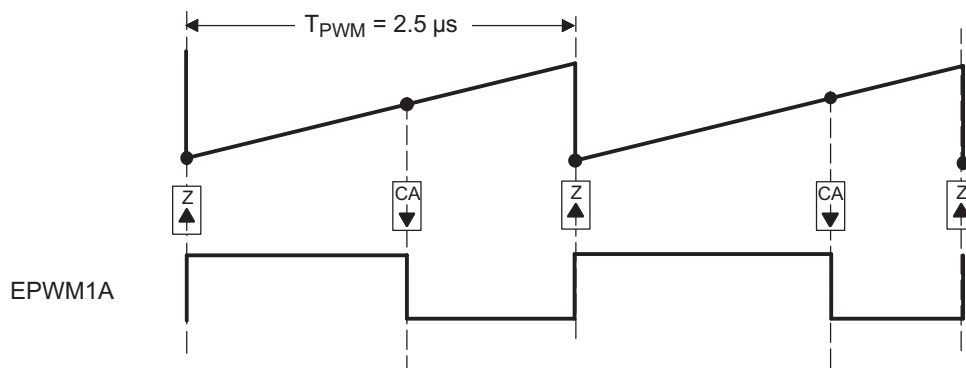
```

**18.15.1.8.2 Implementing a DAC function Using an R+C Reconstruction Filter**

In this example, the PWM requirements are:

- PWM frequency = 400 kHz (that is, TBPRD = 250)
- PWM mode = Asymmetrical, Up-count
- Resolution = 14 bits (MEP step size = 150 ps)

Figure 18-92 and Figure 18-93 show the DAC function and the required PWM waveform. As explained previously, configuration for the ePWM1 module is almost identical to the normal case except that the appropriate MEP options need to be enabled/selected.

**Figure 18-92. Simple Reconstruction Filter for a PWM-based DAC**

**Figure 18-93. PWM Waveform Generated for the PWM DAC Function**


The example code shown consists of two main parts:

- Initialization code (executed once)
- Run time code (typically executed within an ISR)

This example assumes a typical MEP\_SP and does not use the SFO library.

[Example 18-5](#) shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

### **Example 18-5. PWM DAC Function Initialization Code**

```

void HrPwmDacDrvCnf(void)
{
    // Config for conventional PWM first
    EPwm1Regs.TBCTL.bit.PRDL = TB_IMMEDIATE;           // Set Immediate load
    EPwm1Regs.TBPRD = 250;                             // Period set for 400 kHz PWM
    hrDAC_period = 250;                                // Used for Q15 to Q0 scaling
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;           // EPWM1 is the Master
    EPwm1Regs.TBCTL.bit.SYNCSEL = TB_SYNC_DISABLE;
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
    EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
    // Note: ChB is initialized here only for comparison purposes, it is not required

    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;     // optional
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;     // optional

    EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
    EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
    EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;               // optional
    EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;             // optional
    // Now configure the HRPWM resources
    EALLOW;                                           // Note these registers are protected
                                                    // and act only on ChA.
    EPwm1Regs.HRCNFG.all = 0x0; // Clear all bits first
    EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;           // Control falling edge position
    EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;           // CMPAHR controls the MEP.
    EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;       // Shadow load on CTR=Zero.
    EDIS;
    MEP_ScaleFactor = 66*256;                          // Start with typical Scale Factor
                                                    // value for 100 MHz.
                                                    // Use SFO functions to update MEP_ScaleFactor
                                                    // dynamically.
}

```

[Example 18-6](#) shows an assembly example of run-time code that can execute in a high-speed ISR loop.

**Example 18-6. PWM DAC Function Run-Time Code**

```

EPWM1_BASE .set 0x6800
CMPAHR1 .set EPWM1_BASE+0x8
;=====
HRPWM_DAC_DRV; (can execute within an ISR or loop)
;=====
    MOVW DP, #_HRDAC_In
    MOVL XAR2,@_HRDAC_In          ; Pointer to input Q15 duty (XAR2)
    MOVL XAR3,#CMPAHR1           ; Pointer to HRPWM CMPA reg (XAR3)

; Output for EPWM1A (HRPWM
    MOV T,*XAR2                  ; T <= duty
    MPY ACC,T,@_hrDAC_period     ; Q15 to Q0 scaling based on period
    ADD ACC,@_hrDAC_period<<15 ; Offset for bipolar operation
    MOV T,@_MEP_ScaleFactor      ; MEP scale factor (from optimizer s/w)
    MPYU P,T,@AL                 ; P <= T * AL, optimizer scaling
    MOVH @AL,P                   ; AL <= P, move result back to ACC
    ADD ACC, #0x080              ; MEP range and rounding adjustment
    MOVL *XAR3,ACC               ; CMPA:CMPAHR(31:8) <= ACC

; Output for EPWM1B (Regular Res) Optional - for comparison purpose only
    MOV *+XAR3[2],AH             ; Store ACCH to regular CMPB

```

### 18.15.2 Appendix A: SFO Library Software - SFO\_TI\_Build\_V8.lib

The following table lists several features of the SFO\_TI\_Build\_V8.lib library.

**Table 18-19. SFO Library Features**

	<b>SFO_TI_Build_V8.lib</b>	<b>Unit</b>
Completion-checking?	Yes	Function return value
Typical cycles required for SFO() to update MEP_ScaleFactor if called repetitively without interrupts	130,000	EPWMCLK cycles

A functional description of the SFO library routine, SFO(), is found below.

#### 18.15.2.1 Scale Factor Optimizer Function - int SFO()

This routine drives the micro-edge positioner (MEP) calibration module to run SFO diagnostics and determine the appropriate MEP scale factor (number of MEP steps per coarse EPWMCLK step) for a device at any given time.

If EPWMCLK = TBCLK = 100 MHz and assuming the MEP step size is 150 ps, the typical scale factor value at 100 MHz = 66 MEP steps per TBCLK unit (10 ns)

The function returns a MEP scale factor value:

$$\text{MEP\_ScaleFactor} = \text{Number of MEP steps per EPWMCLK.}$$

#### Constraints when using this function:

- SFO() can be used with a minimum EPWMCLK = TBCLK = 50 MHz. MEP diagnostics logic uses EPWMCLK and not TBCLK, so the EPWMCLK restriction is an important constraint. Below 50 MHz, with device process variation, the MEP step size may decrease under cold temperature and high core voltage conditions to such a point, that 255 MEP steps will not span an entire EPWMCLK cycle.
- At any time, SFO() can be called to run SFO diagnostics on the MEP calibration module

#### Usage:

- SFO() can be called at any time in the background while the ePWM channels are running in HRPWM mode. The scale factor result obtained can be applied to all ePWM channels running in HRPWM mode because the function makes use of the diagnostics logic in the MEP calibration module (which runs independently of ePWM channels).
- This routine returns a 1 when calibration is finished and a new scale factor has been calculated, or a 0 if calibration is still running. The routine returns a 2 if there is an error, and the MEP\_ScaleFactor is greater than the maximum 255 fine steps per coarse EPWMCLK cycle. In this case, the HRMSTEP register will maintain the last MEP scale factor value less than 256 for auto conversion.
- All ePWM modules operating in HRPWM incur only a 3-EPWMCLK cycle minimum duty cycle limitation when high-resolution period control is not used. If high-resolution period control is enabled, there is an additional duty cycle limitation 3-EPWMCLK cycles before the end of the PWM period (see [Section 18.15.1.5.3](#)).
- The SFO() function also updates the HRMSTEP register with the scale factor result. If the HRCNFG[AUTOCONV] bit is set, the application software is responsible only for setting CMPAHR = fraction(PWMduty\*PWMperiod) << 8 or CMPBHR = fraction(PWMduty\*PWMperiod) << 8 or TBPRDHR = fraction(PWMperiod) while running SFO() in the background. The MEP Calibration Module will then use the values in the HRMSTEP and CMPAHR/CMPBHR/TBPRDHR register to automatically calculate the appropriate number of MEP steps represented by the fractional duty cycle or period and move the high-resolution ePWM signal edge accordingly.
- If the HRCNFG[AUTOCONV] bit is clear, the HRMSTEP register is ignored. The application software will need to perform the necessary calculations manually so that:
  - $\text{CMPAHR} = (\text{fraction}(\text{PWMduty} * \text{PWMperiod}) * \text{MEP Scale Factor}) << 8 + 0x080.$
  - Similar behavior applies for TBPHSHR, CMPBHR, DBREDHR, DBFEDHR. Auto-conversion must be enabled when using TBPRDHR.

The routine can be run as a background task in a slow loop requiring negligible CPU cycles. The repetition rate at which an SFO function needs to be executed depends on the application's operating environment. As with all digital CMOS devices, temperature and supply voltage variations have an effect on MEP operation. However, in most applications these parameters vary slowly and therefore it is often sufficient to execute the SFO function once every 5 to 10 seconds. If more rapid variations are expected, then execution may have to be performed more frequently to match the application. Note, there is no high limit restriction on the SFO function repetition rate, hence it can execute as quickly as the background loop is capable.

While using the HRPWM feature, HRPWM logic will not be active for the first three EPWMCLK cycles of the PWM period (and the last three EPWMCLK cycles of the PWM period if TBPRDHR is used). While running the application in this configuration, if high-resolution period control is disabled (HRPCTL[HRPE=0]) and the CMPA/CMPB register value is less than three cycles, then its CMPAHR/CMPBHR register must be cleared to zero. If high-resolution period control is enabled (HRPCTL[HRPE=1]), the CMPA register value must not fall below three or above TBPRD-3. This would avoid any unexpected transitions on the PWM signal.

### 18.15.2.2 Software Usage

The software library function SFO(), calculates the MEP scale factor for the HRPWM-supported ePWM modules. The scale factor is an integer value in the range 1-255, and represents the number of micro step edge positions available for a system clock period. The scale factor value is returned in an integer variable called MEP\_ScaleFactor. For example, see [Table 18-20](#).

**Table 18-20. Factor Values**

Software Function call	Functional Description	Updated Variables
SFO()	Returns MEP scale factor in the HRMSTEP register	MEP_ScaleFactor and HRMSTEP register.

To use the HRPWM feature of the ePWMs, it is recommended that the SFO function be used as described here.

#### Step 1. Add "Include" Files

The SFO\_V8.h file needs to be included as follows. This include file is mandatory while using the SFO library function. For the SFO() to operate, the appropriate (Device)\_Device.h and (Device)\_Epwm\_defines.h must be included in the project. These include files are optional if customized header files are used in the end applications.

#### Example 18-7. A Sample of How to Add "Include" Files

```
#include "F28x7x_Device.h"           // F28x7x Headerfile
#include "F28x7x_EPwm_defines.h"    // init defines
#include "SFO_V8.h"                  // SFO lib functions (needed for HRPWM)
```

#### Step 2. Element Declaration

Declare an integer variable for the scale factor value as shown below.



**Example 18-8. Declaring an Element**

```
int MEP_ScaleFactor = 0;    //scale factor value
volatile struct EPWM_REGS *ePWM[] = {0, &EPwm1Regs, &EPwm2Regs, &EPwm3Regs,
&EPwm4Regs};
```

**Step 3. MEP\_ScaleFactor Initialization**

The SFO() function does not require a starting scale factor value in MEP\_ScaleFactor. Prior to using the MEP\_ScaleFactor variable in application code, SFO() should be called to drive the MEP calibration module to calculate an MEP\_ScaleFactor value.

As part of the one-time initialization code prior to using MEP\_ScaleFactor, include the following:

**Example 18-9. Initializing With a Scale Factor Value**

```
MEP_ScaleFactor initialized using function SFO ()
while (SFO() == 0) {} // MEP_ScaleFactor calculated by MEP Cal Module
```

**Step 4. Application Code**

While the application is running, fluctuations in both device temperature and supply voltage may be expected. To be sure that optimal Scale Factors are used for each ePWM module, the SFO function should be re-run periodically as part of a slower back-ground loop. Some examples of this are shown here.

---

**NOTE:** See the HRPWM\_SFO example in the device-specific C/C++ header files and peripheral examples available from the TI website.

---

**Example 18-10. SFO Function Calls**

```
main ()
{
  int status;
  // User code
  // ePWM1, 2, 3, 4 are running in HRPWM mode
  // The status variable returns 1 once a new MEP_ScaleFactor has been
  // calculated by the MEP Calibration Module running SFO
  // diagnostics.

  status = SFO();
  if(status==2) {ESTOP0;} // The function returns a 2 if MEP_ScaleFactor is greater
                        // than the maximum 255 allowed (error condition)
}
```

## 18.16 ePWM Registers

This section describes the Enhanced Pulse Width Modulator registers.

### 18.16.1 ePWM Base Addresses

**Table 18-21. ePWM Base Address Table**

Device Registers	Register Name	Start Address	End Address
EPwm1Regs	EPWM_REGS	0x0000_4000	0x0000_40FF
EPwm2Regs	EPWM_REGS	0x0000_4100	0x0000_41FF
EPwm3Regs	EPWM_REGS	0x0000_4200	0x0000_42FF
EPwm4Regs	EPWM_REGS	0x0000_4300	0x0000_43FF
EPwm5Regs	EPWM_REGS	0x0000_4400	0x0000_44FF
EPwm6Regs	EPWM_REGS	0x0000_4500	0x0000_45FF
EPwm7Regs	EPWM_REGS	0x0000_4600	0x0000_46FF
EPwm8Regs	EPWM_REGS	0x0000_4700	0x0000_47FF
SyncSocRegs	SYNC_SOC_REGS	0x0000_7940	0x0000_794F

## 18.16.2 EPWM\_REGS Registers

Table 18-22 lists the EPWM\_REGS registers. All register offset addresses not listed in Table 18-22 should be considered as reserved locations and the register contents should not be modified.

**Table 18-22. EPWM\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	TBCTL	Time Base Control Register		<a href="#">Go</a>
1h	TBCTL2	Time Base Control Register 2		<a href="#">Go</a>
4h	TBCTR	Time Base Counter Register		<a href="#">Go</a>
5h	TBSTS	Time Base Status Register		<a href="#">Go</a>
8h	CMPCTL	Counter Compare Control Register		<a href="#">Go</a>
9h	CMPCTL2	Counter Compare Control Register 2		<a href="#">Go</a>
Ch	DBCTL	Dead-Band Generator Control Register		<a href="#">Go</a>
Dh	DBCTL2	Dead-Band Generator Control Register 2		<a href="#">Go</a>
10h	AQCTL	Action Qualifier Control Register		<a href="#">Go</a>
11h	AQTSRCSEL	Action Qualifier Trigger Event Source Select Register		<a href="#">Go</a>
14h	PCCTL	PWM Chopper Control Register		<a href="#">Go</a>
18h	VCAPCTL	Valley Capture Control Register		<a href="#">Go</a>
19h	VCNTCFG	Valley Counter Config Register		<a href="#">Go</a>
20h	HRCNFG	HRPWM Configuration Register	EALLOW	<a href="#">Go</a>
21h	HRPWR	HRPWM Power Register	EALLOW	<a href="#">Go</a>
26h	HRMSTEP	HRPWM MEP Step Register	EALLOW	<a href="#">Go</a>
27h	HRCNFG2	HRPWM Configuration 2 Register	EALLOW	<a href="#">Go</a>
2Dh	HRPCTL	High Resolution Period Control Register	EALLOW	<a href="#">Go</a>
2Eh	TRREM	Translator High Resolution Remainder Register	EALLOW	<a href="#">Go</a>
34h	GLDCTL	Global PWM Load Control Register	EALLOW	<a href="#">Go</a>
35h	GLDCFG	Global PWM Load Config Register	EALLOW	<a href="#">Go</a>
38h	EPWMXLINK	EPWMx Link Register		<a href="#">Go</a>
40h	AQCTLA	Action Qualifier Control Register For Output A		<a href="#">Go</a>
41h	AQCTLA2	Additional Action Qualifier Control Register For Output A		<a href="#">Go</a>
42h	AQCTLB	Action Qualifier Control Register For Output B		<a href="#">Go</a>
43h	AQCTLB2	Additional Action Qualifier Control Register For Output B		<a href="#">Go</a>
47h	AQSFR	Action Qualifier Software Force Register		<a href="#">Go</a>
49h	AQCSFR	Action Qualifier Continuous S/W Force Register		<a href="#">Go</a>
50h	DBREDHR	Dead-Band Generator Rising Edge Delay High Resolution Mirror Register		<a href="#">Go</a>
51h	DBRED	Dead-Band Generator Rising Edge Delay High Resolution Mirror Register		<a href="#">Go</a>
52h	DBFEDHR	Dead-Band Generator Falling Edge Delay High Resolution Register		<a href="#">Go</a>
53h	DBFED	Dead-Band Generator Falling Edge Delay Count Register		<a href="#">Go</a>
60h	TBPHS	Time Base Phase High		<a href="#">Go</a>
62h	TBPRDHR	Time Base Period High Resolution Register		<a href="#">Go</a>
63h	TBPRD	Time Base Period Register		<a href="#">Go</a>
6Ah	CMPA	Counter Compare A Register		<a href="#">Go</a>
6Ch	CMPB	Compare B Register		<a href="#">Go</a>
6Fh	CMPC	Counter Compare C Register		<a href="#">Go</a>
71h	CMPD	Counter Compare D Register		<a href="#">Go</a>

**Table 18-22. EPWM\_REGS Registers (continued)**

Offset	Acronym	Register Name	Write Protection	Section
74h	GLDCTL2	Global PWM Load Control Register 2		<a href="#">Go</a>
77h	SWVDELVAL	Software Valley Mode Delay Register		<a href="#">Go</a>
80h	TZSEL	Trip Zone Select Register	EALLOW	<a href="#">Go</a>
82h	TZDCSEL	Trip Zone Digital Comparator Select Register	EALLOW	<a href="#">Go</a>
84h	TZCTL	Trip Zone Control Register	EALLOW	<a href="#">Go</a>
85h	TZCTL2	Additional Trip Zone Control Register	EALLOW	<a href="#">Go</a>
86h	TZCTLDCA	Trip Zone Control Register Digital Compare A	EALLOW	<a href="#">Go</a>
87h	TZCTLDCB	Trip Zone Control Register Digital Compare B	EALLOW	<a href="#">Go</a>
8Dh	TZEINT	Trip Zone Enable Interrupt Register	EALLOW	<a href="#">Go</a>
93h	TZFLG	Trip Zone Flag Register		<a href="#">Go</a>
94h	TZCBCFLG	Trip Zone CBC Flag Register		<a href="#">Go</a>
95h	TZOSTFLG	Trip Zone OST Flag Register		<a href="#">Go</a>
97h	TZCLR	Trip Zone Clear Register	EALLOW	<a href="#">Go</a>
98h	TZCBCCLR	Trip Zone CBC Clear Register	EALLOW	<a href="#">Go</a>
99h	TZOSTCLR	Trip Zone OST Clear Register	EALLOW	<a href="#">Go</a>
9Bh	TZFRC	Trip Zone Force Register	EALLOW	<a href="#">Go</a>
A4h	ETSEL	Event Trigger Selection Register		<a href="#">Go</a>
A6h	ETPS	Event Trigger Pre-Scale Register		<a href="#">Go</a>
A8h	ETFLG	Event Trigger Flag Register		<a href="#">Go</a>
AAh	ETCLR	Event Trigger Clear Register		<a href="#">Go</a>
ACh	ETFRC	Event Trigger Force Register		<a href="#">Go</a>
A Eh	ETINTPS	Event-Trigger Interrupt Pre-Scale Register		<a href="#">Go</a>
B0h	ETSOCPS	Event-Trigger SOC Pre-Scale Register		<a href="#">Go</a>
B2h	ETCNTINITCTL	Event-Trigger Counter Initialization Control Register		<a href="#">Go</a>
B4h	ETCNTINIT	Event-Trigger Counter Initialization Register		<a href="#">Go</a>
C0h	DCTRIPSEL	Digital Compare Trip Select Register	EALLOW	<a href="#">Go</a>
C3h	DCACTL	Digital Compare A Control Register	EALLOW	<a href="#">Go</a>
C4h	DCBCTL	Digital Compare B Control Register	EALLOW	<a href="#">Go</a>
C7h	DCFCTL	Digital Compare Filter Control Register	EALLOW	<a href="#">Go</a>
C8h	DCCAPCTL	Digital Compare Capture Control Register	EALLOW	<a href="#">Go</a>
C9h	DCFOFFSET	Digital Compare Filter Offset Register		<a href="#">Go</a>
CAh	DCFOFFSETCNT	Digital Compare Filter Offset Counter Register		<a href="#">Go</a>
CBh	DCFWINDOW	Digital Compare Filter Window Register		<a href="#">Go</a>
CCh	DCFWINDOWCNT	Digital Compare Filter Window Counter Register		<a href="#">Go</a>
CFh	DCCAP	Digital Compare Counter Capture Register		<a href="#">Go</a>
D2h	DCAHTRIPSEL	Digital Compare AH Trip Select	EALLOW	<a href="#">Go</a>
D3h	DCALTRIPSEL	Digital Compare AL Trip Select	EALLOW	<a href="#">Go</a>
D4h	DCBHTRIPSEL	Digital Compare BH Trip Select	EALLOW	<a href="#">Go</a>
D5h	DCBLTRIPSEL	Digital Compare BL Trip Select	EALLOW	<a href="#">Go</a>
FAh	EPWMLOCK	EPWM Lock Register		<a href="#">Go</a>
FDh	HWVDELVAL	Hardware Valley Mode Delay Register		<a href="#">Go</a>
FEh	VCNTVAL	Hardware Valley Counter Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 18-23](#) shows the codes that are used for access types in this section.

**Table 18-23. EPWM\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1C	W1C	Write 1 to clear
W1S	W1S	Write 1 to set
WOnce	WOnce	Write Write once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**18.16.2.1 TBCTL Register (Offset = 0h) [reset = 83h]**

TBCTL is shown in [Figure 18-94](#) and described in [Table 18-24](#).

Return to the [Summary Table](#).

Time Base Control Register

**Figure 18-94. TBCTL Register**

15		14		13		12		11		10		9		8	
FREE_SOFT				PHSDIR		CLKDIV				HSPCLKDIV					
R/W-0h				R/W-0h		R/W-0h				R/W-1h					
7		6		5		4		3		2		1		0	
HSPCLKDIV		SWFSYNC		SYNCOSSEL				PRDLD		PHSEN		CTRMODE			
R/W-1h		R-0/W1S-0h		R/W-0h				R/W-0h		R/W-0h		R/W-3h			

**Table 18-24. TBCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events  00: Stop after the next time-base counter increment or decrement 01: Stop when counter completes a whole cycle: - Up-count mode: stop when the time-base counter = period (TBCTR = TBPRD) - Down-count mode: stop when the time-base counter = 0x00 (TBCTR = 0x00) - Up-down-count mode: stop when the time-base counter = 0x00 (TBCTR = 0x00) 1x: Free run Reset type: SYSRSn
13	PHSDIR	R/W	0h	Phase Direction Bit  This bit is only used when the time-base counter is configured in the up-down-count mode. The PHSDIR bit indicates the direction the time-base counter (TBCTR) will count after a synchronization event occurs and a new phase value is loaded from the phase (TBPHS) register. This is irrespective of the direction of the counter before the synchronization event..  In the up-count and down-count modes this bit is ignored.  0: Count down after the synchronization event. 1: Count up after the synchronization event. Reset type: SYSRSn

**Table 18-24. TBCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12-10	CLKDIV	R/W	0h	<p>Time Base Clock Pre-Scale Bits</p> <p>These bits select the time base clock pre-scale value (<math>TBCLK = EPWMCLK / (HSPCLKDIV * CLKDIV)</math>):</p> <p>000: /1 (default on reset)            001: /2            010: /4            011: /8            100: /16            101: /32            110: /64            111: /128</p> <p>Reset type: SYSRSn</p>
9-7	HSPCLKDIV	R/W	1h	<p>High Speed Time Base Clock Pre-Scale Bits</p> <p>These bits determine part of the time-base clock prescale value. <math>TBCLK = EPWMCLK / (HSPCLKDIV * CLKDIV)</math>. This divisor emulates the HSPCLK in the TMS320x281x system as used on the Event Manager (EV) peripheral.</p> <p>000: /1            001: /2 (default on reset)            010: /4            011: /6            100: /8            101: /10            110: /12            111: /14</p> <p>Reset type: SYSRSn</p>
6	SWFSYNC	R-0/W1S	0h	<p>Software Forced Sync Pulse</p> <p>0: Writing a 0 has no effect and reads always return a 0.            1: Writing a 1 forces a one-time synchronization pulse to be generated.</p> <p>SWFSYNC affects EPWMxSYNCO only when SYNCOSSEL = 00.</p> <p>Reset type: SYSRSn</p>
5-4	SYNCOSSEL	R/W	0h	<p>Sync Output Select</p> <p>00: EPWMxSYNCO / SWFSYNC            01: CTR = zero: Time-base counter equal to zero (<math>TBCTR = 0x00</math>)            10: CTR = CMPB : Time-base counter equal to counter-compare B (<math>TBCTR = CMPB</math>)            11: EPWMxSYNCO is defined by <math>TBCTL2[SYNCOSSELX]</math></p> <p>Reset type: SYSRSn</p>
3	PRDL	R/W	0h	<p>Active Period Reg Load from Shadow Select</p> <p>0: The period register (TBPRD) is loaded from its shadow register when the time-base counter, TBCTR, is equal to zero and/or a sync event as determined by the <math>TBCTL2[PRDLDSYNC]</math> bit.            A write/read to the TBPRD register accesses the shadow register.</p> <p>1: Immediate Mode (Shadow register bypassed): A write or read to the TBPRD register accesses the active register.</p> <p>Reset type: SYSRSn</p>

**Table 18-24. TBCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	PHSEN	R/W	0h	<p>Counter Reg Load from Phase Reg Enable</p> <p>0: Do not load the time-base counter (TBCTR) from the time-base phase register (TBPHS).</p> <p>1: Allow Counter to be loaded from the Phase register (TBPHS) and shadow to active load events when an EPWMxSYNCl input signal occurs or a software-forced sync signal, see bit 6.</p> <p>Reset type: SYSRSn</p>
1-0	CTRMODE	R/W	3h	<p>Counter Mode</p> <p>The time-base counter mode is normally configured once and not changed during normal operation. If you change the mode of the counter, the change will take effect at the next TBCLK edge and the current counter value shall increment or decrement from the value before the mode change. These bits set the time-base counter mode of operation as follows:</p> <p>00: Up-count mode</p> <p>01: Down-count mode</p> <p>10: Up-down count mode</p> <p>11: Freeze counter operation (default on reset)</p> <p>Reset type: SYSRSn</p>



**18.16.2.2 TBCTL2 Register (Offset = 1h) [reset = 0h]**

TBCTL2 is shown in [Figure 18-95](#) and described in [Table 18-25](#).

Return to the [Summary Table](#).

Time Base Control Register 2

**Figure 18-95. TBCTL2 Register**

15		14		13		12		11		10		9		8	
PRDLDSYNC				SYNCOSELX				RESERVED							
R/W-0h				R/W-0h				R-0-0h							
7		6		5		4		3		2		1		0	
OSHTSYNC		OSHTSYNCMODE		RESERVED		RESERVED									
R-0/W1S-0h		R/W-0h						R-0-0h							

**Table 18-25. TBCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	PRDLDSYNC	R/W	0h	Shadow to Active Period Register Load on SYNC event  00: Shadow to Active Load of TBPRD occurs only when TBCTR = 0 (same as legacy). 01: Shadow to Active Load of TBPRD occurs both when TBCTR = 0 and when SYNC occurs. 10: Shadow to Active Load of TBPRD occurs only when a SYNC is received. 11: Reserved  Note: This bit selection is valid only if TBCTL[PRDLD]=0. Reset type: SYSRSn
13-12	SYNCOSELX	R/W	0h	Extended selection bits for SYNCOUT  00: Disabled EPWMxSYNCO sync signal 01: EPWMxSYNCO = CMPC 10: EPWMxSYNCO = CMPD 11: Reserved Reset type: SYSRSn
11-8	RESERVED	R-0	0h	Reserved
7	OSHTSYNC	R-0/W1S	0h	Oneshot sync bit  0: Writing a '0' has no effect. 1: Allow one sync pulse to propagate. Reset type: SYSRSn
6	OSHTSYNCMODE	R/W	0h	Oneshot sync enable bit  0: Oneshot sync mode disabled 1: Oneshot sync mode enabled Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4-0	RESERVED	R-0	0h	Reserved

**18.16.2.3 TBCTR Register (Offset = 4h) [reset = 0h]**

TBCTR is shown in [Figure 18-96](#) and described in [Table 18-26](#).

Return to the [Summary Table](#).

Time Base Counter Register

**Figure 18-96. TBCTR Register**

15	14	13	12	11	10	9	8
TBCTR							
R/W-0h							
7	6	5	4	3	2	1	0
TBCTR							
R/W-0h							

**Table 18-26. TBCTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TBCTR	R/W	0h	Time Base Counter Register Reset type: SYSRSn

**18.16.2.4 TBSTS Register (Offset = 5h) [reset = 1h]**

TBSTS is shown in [Figure 18-97](#) and described in [Table 18-27](#).

Return to the [Summary Table](#).

Time Base Status Register

**Figure 18-97. TBSTS Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED					CTRMAX	SYNCI	CTDIR
R-0-0h					R/W1C-0h	R/W1C-0h	R-1h

**Table 18-27. TBSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R-0	0h	Reserved
2	CTRMAX	R/W1C	0h	Time-Base Counter Max Latched Status Bit  0: Reading a 0 indicates the time-base counter never reached its maximum value. Writing a 0 will have no effect. 1: Reading a 1 on this bit indicates that the time-base counter reached the max value 0xFFFF. Writing a 1 to this bit will clear the latched event. Reset type: SYSRSn
1	SYNCI	R/W1C	0h	Input Synchronization Latched Status Bit  0: Writing a 0 will have no effect. Reading a 0 indicates no external synchronization event has occurred. 1: Reading a 1 on this bit indicates that an external synchronization event has occurred (EPWMxSYNCI). Writing a 1 to this bit will clear the latched event. Reset type: SYSRSn
0	CTDIR	R	1h	Time Base Counter Direction Status Bit  0: Time-Base Counter is currently counting down. 1: Time-Base Counter is currently counting up.  Note: This bit is only valid when the counter is not frozen. Reset type: SYSRSn

**18.16.2.5 CMPCTL Register (Offset = 8h) [reset = 0h]**

CMPCTL is shown in [Figure 18-98](#) and described in [Table 18-28](#).

Return to the [Summary Table](#).

Counter Compare Control Register

**Figure 18-98. CMPCTL Register**

15	14	13	12	11	10	9	8
RESERVED		LOADBSYNC		LOADASYNC		SHDWBFULL	SHDWAFULL
R-0-0h		R/W-0h		R/W-0h		R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	SHDWBMODE	RESERVED	SHDWAMODE	LOADBMODE		LOADAMODE	
R-0-0h	R/W-0h	R-0-0h	R/W-0h	R/W-0h		R/W-0h	

**Table 18-28. CMPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R-0	0h	Reserved
13-12	LOADBSYNC	R/W	0h	Shadow to Active CMPB Register Load on SYNC event  00: Shadow to Active Load of CMPB:CMPBHR occurs according to LOADBMODE (bits 1,0) (same as legacy) 01: Shadow to Active Load of CMPB:CMPBHR occurs both according to LOADBMODE bits and when SYNC occurs 10: Shadow to Active Load of CMPB:CMPBHR occurs only when a SYNC is received 11: Reserved  Note: This bit is valid only if CMPCTL[SHDWBMODE] = 0. Reset type: SYSRSn
11-10	LOADASYNC	R/W	0h	Shadow to Active CMPA Register Load on SYNC event  00: Shadow to Active Load of CMPA:CMPAHR occurs according to LOADAMODE (bits 1,0) (same as legacy) 01: Shadow to Active Load of CMPA:CMPAHR occurs both according to LOADAMODE bits and when SYNC occurs 10: Shadow to Active Load of CMPA:CMPAHR occurs only when a SYNC is received 11: Reserved  Note: This bit is valid only if CMPCTL[SHDWAMODE] = 0. Reset type: SYSRSn
9	SHDWBFULL	R	0h	Counter-compare B (CMPB) Shadow Register Full Status Flag  This bit self clears once a loadstrobe occurs.  0: CMPB shadow FIFO not full yet 1: Indicates the CMPB shadow FIFO is full a CPU write will overwrite current shadow value Reset type: SYSRSn
8	SHDWAFULL	R	0h	Counter-compare A (CMPA) Shadow Register Full Status Flag  The flag bit is set when a 32-bit write to CMPA:CMPAHR register or a 16-bit write to CMPA register is made. A 16-bit write to CMPAHR register will not affect the flag. This bit self clears once a load-strobe occurs.  0: CMPA shadow FIFO not full yet 1: Indicates the CMPA shadow FIFO is full, a CPU write will overwrite the current shadow value Reset type: SYSRSn

**Table 18-28. CMPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	RESERVED	R-0	0h	Reserved
6	SHDWBMODE	R/W	0h	Counter-compare B (CMPB) Register Operating Mode 0: Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register 1: Immediate mode. Only the active compare B register is used. All writes and reads directly access the active register for immediate compare action Reset type: SYSRSn
5	RESERVED	R-0	0h	Reserved
4	SHDWAMODE	R/W	0h	Counter-compare A (CMPA) Register Operating Mode 0: Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register 1: Immediate mode. Only the active compare register is used. All writes and reads directly access the active register for immediate compare action Reset type: SYSRSn
3-2	LOADBMODE	R/W	0h	Active Counter-Compare B (CMPB) Load From Shadow Select Mode This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1). 00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible) Reset type: SYSRSn
1-0	LOADAMODE	R/W	0h	Active Counter-Compare A (CMPA) Load From Shadow Select Mode This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1). 00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible) Reset type: SYSRSn

**18.16.2.6 CMPCTL2 Register (Offset = 9h) [reset = 0h]**

CMPCTL2 is shown in [Figure 18-99](#) and described in [Table 18-29](#).

Return to the [Summary Table](#).

Counter Compare Control Register 2

**Figure 18-99. CMPCTL2 Register**

15	14	13	12	11	10	9	8
RESERVED		LOADDSYNC		LOADCSYNC		RESERVED	
R-0-0h		R/W-0h		R/W-0h		R-0-0h	
7	6	5	4	3	2	1	0
RESERVED	SHDWDMODE	RESERVED	SHDWCMODE	LOADDMODE		LOADCMODE	
R-0-0h	R/W-0h	R-0-0h	R/W-0h	R/W-0h		R/W-0h	

**Table 18-29. CMPCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R-0	0h	Reserved
13-12	LOADDSYNC	R/W	0h	Shadow to Active CMPD Register Load on SYNC event  00: Shadow to Active Load of CMPD occurs according to LOADDMODE 01: Shadow to Active Load of CMPD occurs both according to LOADDMODE bits and when SYNC occurs 10: Shadow to Active Load of CMPD occurs only when a SYNC is received 11: Reserved  Note: This bit is valid only if CMPCTL2[SHDWDMODE] = 0. Reset type: SYSRSn
11-10	LOADCSYNC	R/W	0h	Shadow to Active CMPC Register Load on SYNC event  00: Shadow to Active Load of CMPC occurs according to LOADCMODE 01: Shadow to Active Load of CMPC occurs both according to LOADCMODE bits and when SYNC occurs 10: Shadow to Active Load of CMPC occurs only when a SYNC is received 11: Reserved  Note: This bit is valid only if CMPCTL2[SHDWCMODE] = 0. Reset type: SYSRSn
9-7	RESERVED	R-0	0h	Reserved
6	SHDWDMODE	R/W	0h	Counter-Compare D Register Operating Mode  0: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register. 1: Immediate mode - only the Active compare register is used. All writes/reads via the CPU directly access the Active register for immediate Compare action. Reset type: SYSRSn
5	RESERVED	R-0	0h	Reserved

**Table 18-29. CMPCTL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	SHDWCMODE	R/W	0h	Counter-Compare C Register Operating Mode  0: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register. 1: Immediate mode - only the Active compare register is used. All writes/reads via the CPU directly access the Active register for immediate Compare action. Reset type: SYSRSn
3-2	LOADDMODE	R/W	0h	Active Counter-Compare D (CMPD) Load from Shadow Select Mode  00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible)  Note: Has no effect in Immediate mode. Reset type: SYSRSn
1-0	LOADCMODE	R/W	0h	Active Counter-Compare C (CMPC) Load from Shadow Select Mode  00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible)  Note: Has no effect in Immediate mode. Reset type: SYSRSn

**18.16.2.7 DBCTL Register (Offset = Ch) [reset = 0h]**

 DBCTL is shown in [Figure 18-100](#) and described in [Table 18-30](#).

 Return to the [Summary Table](#).

Dead-Band Generator Control Register

**Figure 18-100. DBCTL Register**

15		14		13		12		11		10		9		8	
HALFCYCLE		DEDB_MODE		OUTSWAP		SHDWDBFED MODE		SHDWDBRED MODE		LOADFEDMODE					
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	
LOADREDMODE				IN_MODE				POLSEL				OUT_MODE			
R/W-0h				R/W-0h				R/W-0h				R/W-0h		R/W-0h	

**Table 18-30. DBCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	HALFCYCLE	R/W	0h	Half Cycle Clocking Enable Bit  0: Full cycle clocking enabled. The dead-band counters are clocked at the TBCLK rate. 1: Half cycle clocking enabled. The dead-band counters are clocked at TBCLK*2. Reset type: SYSRSn
14	DEDB_MODE	R/W	0h	Dead Band Dual-Edge B Mode Control (S8 switch)  0: Rising edge delay applied to InA/InB as selected by S4 switch (INMODE bits) on A signal path only. Falling edge delay applied to InA/InB as selected by S5 switch (INMODE bits) on B signal path only. 1: Rising edge delay and falling edge delay applied to source selected by S4 switch (INMODE bits) and output to B signal path only. Note: When this bit is set to 1, user should always either set OUT_MODE bits such that Apath = InA OR OUTSWAP bits such that OutA=Bpath otherwise, OutA will be invalid. Reset type: SYSRSn
13-12	OUTSWAP	R/W	0h	Dead Band Output Swap Control  Bit 13 controls the S6 switch and bit 12 controls the S7 switch.  00: OutA and OutB signals are as defined by OUT-MODE bits. 01: OutA = A-path as defined by OUT-MODE bits. OutB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A signal path). 10: OutA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B signal path). OutB = B-path as defined by OUT-MODE bits. 11: OutA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B signal path). OutB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A signal path). Reset type: SYSRSn



**Table 18-30. DBCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	SHDWDBFEDMODE	R/W	0h	<p>FED Dead-Band Load Mode</p> <p>0: Immediate mode. Only the active DBFED register is used. All writes/reads via the CPU directly access the active register for immediate "FED dead-band action."</p> <p>1: Shadow mode. Operates as a double buffer. All writes via the CPU access Shadow register. Default at Reset is Immediate mode (for compatibility with legacy).</p> <p>Reset type: SYSRSn</p>
10	SHDWDBREDDMODE	R/W	0h	<p>RED Dead-Band Load Mode</p> <p>0: Immediate mode. Only the active DBRED register is used. All writes/reads via the CPU directly access the active register for immediate "RED dead-band action."</p> <p>1: Shadow mode. Operates as a double buffer. All writes via the CPU access Shadow register. Default at Reset is Immediate mode (for compatibility with legacy).</p> <p>Reset type: SYSRSn</p>
9-8	LOADFEDMODE	R/W	0h	<p>Active DBFED Load from Shadow Select Mode</p> <p>00: Load on Counter = 0 (CNT_eq)</p> <p>01: Load on Counter = Period (PRD_eq)</p> <p>10: Load on either Counter = 0, or Counter = Period</p> <p>11: Freeze (no loads possible)</p> <p>Note: has no effect in Immediate mode.</p> <p>Reset type: SYSRSn</p>
7-6	LOADREDDMODE	R/W	0h	<p>Active DBRED Load from Shadow Select Mode</p> <p>00: Load on Counter = 0 (CNT_eq)</p> <p>01: Load on Counter = Period (PRD_eq)</p> <p>10: Load on either Counter = 0, or Counter = Period</p> <p>11: Freeze (no loads possible)</p> <p>Note: has no effect in Immediate mode.</p> <p>Reset type: SYSRSn</p>
5-4	IN_MODE	R/W	0h	<p>Dead-Band Input Mode Control</p> <p>Bit 5 controls the S5 switch and bit 4 controls the S4 switch shown. This allows you to select the input source to the falling-edge and rising-edge delay. To produce classical dead-band waveforms the default is EPWMxA In is the source for both falling and rising-edge delays.</p> <p>00: EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.</p> <p>01: EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal.</p> <p>EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal.</p> <p>10: EPWMxA In (from the action-qualifier) is the source for rising-edge delayed signal.</p> <p>EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal.</p> <p>11: EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.</p> <p>Reset type: SYSRSn</p>

**Table 18-30. DBCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	POLSEL	R/W	0h	<p>Polarity Select Control</p> <p>Bit 3 controls the S3 switch and bit 2 controls the S2 switch. This allows you to selectively invert one of the delayed signals before it is sent out of the dead-band submodule. The following descriptions correspond to classical upper/lower switch control as found in one leg of a digital motor control inverter. These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0x0. Other enhanced modes are also possible, but not regarded as typical usage modes.</p> <p>00: Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).</p> <p>01: Active low complementary (ALC) mode. EPWMxA is inverted.</p> <p>10: Active high complementary (AHC). EPWMxB is inverted.</p> <p>11: Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.</p> <p>Reset type: SYSRSn</p>
1-0	OUT_MODE	R/W	0h	<p>Dead-Band Output Mode Control</p> <p>Bit 1 controls the S1 switch and bit 0 controls the S0 switch.</p> <p>00: DBM is fully disabled or by-passed. In this mode the POLSEL and IN-MODE bits have no effect.</p> <p>01: Apath = InA (delay is by-passed for A signal path) Bpath = FED (Falling Edge Delay in B signal path)</p> <p>10: Apath = RED (Rising Edge Delay in A signal path) Bpath = InB (delay is by-passed for B signal path)</p> <p>11: DBM is fully enabled (i.e. both RED and FED active)</p> <p>Reset type: SYSRSn</p>

**18.16.2.8 DBCTL2 Register (Offset = Dh) [reset = 0h]**

DBCTL2 is shown in [Figure 18-101](#) and described in [Table 18-31](#).

Return to the [Summary Table](#).

Dead-Band Generator Control Register 2

**Figure 18-101. DBCTL2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED					SHDWDBCTL MODE	LOADDBCTLMODE	
R-0-0h					R/W-0h	R/W-0h	

**Table 18-31. DBCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R-0	0h	Reserved
2	SHDWDBCTLMODE	R/W	0h	DBCTL Load Mode  0: Immediate mode - only the Active DBCTL register is used. All writes/reads via the CPU directly access the Active register. 1: Shadow mode - All writes and reads via the CPU access the Shadow register. Reset type: SYSRSn
1-0	LOADDBCTLMODE	R/W	0h	Active DBCTL Load from Shadow Select Mode  00: Load on Counter = 0 (CNT_eq) 01: Load on Counter = Period (PRD_eq) 10: Load on either Counter = 0, or Counter = Period 11: Freeze (no loads possible)  Note: has no effect in Immediate mode Reset type: SYSRSn

**18.16.2.9 AQCTL Register (Offset = 10h) [reset = 0h]**

AQCTL is shown in [Figure 18-102](#) and described in [Table 18-32](#).

Return to the [Summary Table](#).

Action Qualifier Control Register

**Figure 18-102. AQCTL Register**

15	14	13	12	11	10	9	8
RESERVED				LDAQBSYNC		LDAQASYNC	
R-0-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
RESERVED	SHDWAQBMODE	RESERVED	SHDWAQAMODE	LDAQBMODE		LDAQAMODE	
R-0-0h	R/W-0h	R-0-0h	R/W-0h	R/W-0h		R/W-0h	

**Table 18-32. AQCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-10	LDAQBSYNC	R/W	0h	Shadow to Active AQCTLB Register Load on SYNC event  00: Shadow to Active Load of AQCTLB occurs according to LDAQBMODE 01: Shadow to Active Load of AQCTLB occurs both according to LDAQBMODE bits and when SYNC occurs. 10: Shadow to Active Load of AQCTLB occurs only when a SYNC is received. 11: Reserved  Note: This bit is valid only if AQCTLR[SHDWAQBMODE] = 1. Reset type: SYSRSn
9-8	LDAQASYNC	R/W	0h	Shadow to Active AQCTLA Register Load on SYNC event  00: Shadow to Active Load of AQCTLA occurs according to LDAQAMODE 01: Shadow to Active Load of AQCTLA occurs both according to LDAQAMODE bits and when SYNC occurs. 10: Shadow to Active Load of AQCTLA occurs only when a SYNC is received. 11: Reserved  Note: This bit is valid only if AQCTLR[SHDWAQAMODE] = 1. Reset type: SYSRSn
7	RESERVED	R-0	0h	Reserved
6	SHDWAQBMODE	R/W	0h	Action Qualifier B Register operating mode  1: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register. 0: Immediate mode - only the Active action qualifier register is used. All writes/reads via the CPU directly access the Active register. Reset type: SYSRSn
5	RESERVED	R-0	0h	Reserved
4	SHDWAQAMODE	R/W	0h	Action Qualifier A Register operating mode  1: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register. 0: Immediate mode - only the Active action qualifier register is used. All writes/reads via the CPU directly access the Active register. Reset type: SYSRSn

**Table 18-32. AQCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	LDAQBMODE	R/W	0h	Active Action Qualifier B Load from Shadow Select Mode  00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible)  Note: has no effect in Immediate mode. Reset type: SYSRSn
1-0	LDAQAMODE	R/W	0h	Active Action Qualifier A Load from Shadow Select Mode  00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible)  Note: has no effect in Immediate mode. Reset type: SYSRSn

**18.16.2.10 AQTSRCSEL Register (Offset = 11h) [reset = 0h]**

 AQTSRCSEL is shown in [Figure 18-103](#) and described in [Table 18-33](#).

 Return to the [Summary Table](#).

Action Qualifier Trigger Event Source Select Register

**Figure 18-103. AQTSRCSEL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
T2SEL				T1SEL			
R/W-0h				R/W-0h			

**Table 18-33. AQTSRCSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-4	T2SEL	R/W	0h	T2 Event Source Select Bits  0000: DCAEVT1 0001: DCAEVT2 0010: DCBEVT1 0011: DCBEVT2 0100: TZ1 0101: TZ2 0110: TZ3 0111: EPWMxSYNCl 1000: DCEVTFILT Others: Reserved Reset type: SYSRSn
3-0	T1SEL	R/W	0h	T1 Event Source Select Bits  0000: DCAEVT1 0001: DCAEVT2 0010: DCBEVT1 0011: DCBEVT2 0100: TZ1 0101: TZ2 0110: TZ3 0111: EPWMxSYNCl 1000: DCEVTFILT Others: Reserved Reset type: SYSRSn

**18.16.2.11 PCCTL Register (Offset = 14h) [reset = 0h]**

PCCTL is shown in [Figure 18-104](#) and described in [Table 18-34](#).

Return to the [Summary Table](#).

PWM Chopper Control Register

**Figure 18-104. PCCTL Register**

15	14	13	12	11	10	9	8
RESERVED						CHPDUTY	
R-0-0h						R/W-0h	
7	6	5	4	3	2	1	0
CHPFREQ			OSHTWTH			CHPEN	
R/W-0h			R/W-0h			R/W-0h	

**Table 18-34. PCCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R-0	0h	Reserved
10-8	CHPDUTY	R/W	0h	Chopping Clock Duty Cycle  000: Duty = 1/8 (12.5%) 001: Duty = 2/8 (25.0%) 010: Duty = 3/8 (37.5%) 011: Duty = 4/8 (50.0%) 100: Duty = 5/8 (62.5%) 101: Duty = 6/8 (75.0%) 110: Duty = 7/8 (87.5%) 111: Reserved Reset type: SYSRSn
7-5	CHPFREQ	R/W	0h	Chopping Clock Frequency  000: Divide by 1 (no prescale, = 12.5 MHz at 100 MHz TBCLK) 001: Divide by 2 (6.25 MHz at 100 MHz TBCLK) 010: Divide by 3 (4.16 MHz at 100 MHz TBCLK) 011: Divide by 4 (3.12 MHz at 100 MHz TBCLK) 100: Divide by 5 (2.50 MHz at 100 MHz TBCLK) 101: Divide by 6 (2.08 MHz at 100 MHz TBCLK) 110: Divide by 7 (1.78 MHz at 100 MHz TBCLK) 111: Divide by 8 (1.56 MHz at 100 MHz TBCLK) Reset type: SYSRSn

**Table 18-34. PCCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-1	OSHTWTH	R/W	0h	<p>One-Shot Pulse Width</p> <p>0000: 1 x EPWMCLK / 8 wide (= 80 ns at 100 MHz EPWMCLK)            0001: 2 x EPWMCLK / 8 wide (= 160 ns at 100 MHz EPWMCLK)            0010: 3 x EPWMCLK / 8 wide (= 240 ns at 100 MHz EPWMCLK)            0011: 4 x EPWMCLK / 8 wide (= 320 ns at 100 MHz EPWMCLK)            0100: 5 x EPWMCLK / 8 wide (= 400 ns at 100 MHz EPWMCLK)            0101: 6 x EPWMCLK / 8 wide (= 480 ns at 100 MHz EPWMCLK)            0110: 7 x EPWMCLK / 8 wide (= 560 ns at 100 MHz EPWMCLK)            0111: 8 x EPWMCLK / 8 wide (= 640 ns at 100 MHz EPWMCLK)            1000: 9 x EPWMCLK / 8 wide (= 720 ns at 100 MHz EPWMCLK)            1001: 10 x EPWMCLK / 8 wide (= 800 ns at 100 MHz EPWMCLK)            1010: 11 x EPWMCLK / 8 wide (= 880 ns at 100 MHz EPWMCLK)            1011: 12 x EPWMCLK / 8 wide (= 960 ns at 100 MHz EPWMCLK)            1100: 13 x EPWMCLK / 8 wide (= 1040 ns at 100 MHz EPWMCLK)            1101: 14 x EPWMCLK / 8 wide (= 1120 ns at 100 MHz EPWMCLK)            1110: 15 x EPWMCLK / 8 wide (= 1200 ns at 100 MHz EPWMCLK)            1111: 16 x EPWMCLK / 8 wide (= 1280 ns at 100 MHz EPWMCLK)</p> <p>Reset type: SYSRSn</p>
0	CHPEN	R/W	0h	<p>PWM-Chopping Enable</p> <p>0: Disable (bypass) PWM chopping function            1: Enable chopping function</p> <p>Reset type: SYSRSn</p>



**18.16.2.12 VCAPCTL Register (Offset = 18h) [reset = 0h]**

VCAPCTL is shown in [Figure 18-105](#) and described in [Table 18-35](#).

Return to the [Summary Table](#).

Valley Capture Control Register

**Figure 18-105. VCAPCTL Register**

15	14	13	12	11	10	9	8
RESERVED					EDGEFILTDLY SEL	VDELAYDIV	
R-0-0h					R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
VDELAYDIV	RESERVED		TRIGSEL			VCAPSTART	VCAPE
R/W-0h	R-0-0h		R/W-0h			R-0/W1S-0h	R/W-0h

**Table 18-35. VCAPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R-0	0h	Reserved
10	EDGEFILTDLYSEL	R/W	0h	Valley Switching Mode Delay Selection  0: No delay applied to the edge filter output 1: HWDELAYVAL delay applied to the edge filter output Reset type: SYSRSn
9-7	VDELAYDIV	R/W	0h	Valley Delay Mode Divide Enable  000: HWVDELVAL = SWVDELVAL 001: HWVDELVAL = VCNTVAL+SWVDELVAL 010: HWVDELVAL = VCNTVAL>>1+SWVDELVAL 011: HWVDELVAL = VCNTVAL>>2+SWVDELVAL 100: HWVDELVAL = VCNTVAL>>4+SWVDELVAL  Note: Delay value between the consecutive edge captures can optionally be divided by using these bits. Reset type: SYSRSn
6-5	RESERVED	R-0	0h	Reserved

**Table 18-35. VCAPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-2	TRIGSEL	R/W	0h	<p>Status of Numbered of Captured Events</p> <p>000: Capture sequence is triggered by software via writes to VCAPCTL[VCAPSTART].</p> <p>001: Capture sequence is triggered by CNT_zero event.</p> <p>010: Capture sequence is triggered by PRD_eq event.</p> <p>011: Capture sequence is triggered by CNT_zero or PRD_eq event.</p> <p>100: Capture sequence is triggered by DCAEVT1 event.</p> <p>101: Capture sequence is triggered by DCAEVT2 event.</p> <p>110: Capture sequence is triggered by DCBEVT1 event.</p> <p>111: Capture sequence is triggered by DCBEVT2 event.</p> <p>Note: Valley capture sequence triggered by the selected event in this register field. Once the chosen event occurs the capture sequence is armed. Event captures occur based of the event chosen in DCFCTL[SRCSSEL] register.</p> <p>Note: Same event may not be chosen in both DCFCTL[SRCSSEL] and VCAPCTL[TRIGSEL] registers.</p> <p>Note: Once the chosen event in VCAPCTL[TRIGSEL] occurs, irrespective of the current capture status, capture sequence is retriggered.</p> <p>Reset type: SYSRSn</p>
1	VCAPSTART	R-0/W1S	0h	<p>Valley Capture Start</p> <p>0: Writing a 0 has no effect</p> <p>1: Trigger the capture sequence once if VCAPCTL[TRIGSEL]=0x0</p> <p>Note: This bit is used to start valley capture sequence through software. VCAPCTL[TRIGSEL] has to be chosen for software trigger for this bit to have any effect. Writing of 1 will result in one capture sequence trigger.</p> <p>Reset type: SYSRSn</p>
0	VCAPE	R/W	0h	<p>Valley Capture Enable/Disable</p> <p>0: Disabled</p> <p>1: Enabled</p> <p>Reset type: SYSRSn</p>

**18.16.2.13 VCNTCFG Register (Offset = 19h) [reset = 0h]**

VCNTCFG is shown in [Figure 18-106](#) and described in [Table 18-36](#).

Return to the [Summary Table](#).

Valley Counter Config Register

**Figure 18-106. VCNTCFG Register**

15	14	13	12	11	10	9	8
STOPEDESTS	RESERVED			STOPEDESTS			
R-0h	R-0-0h			R/W-0h			
7	6	5	4	3	2	1	0
STARTEDGESTS	RESERVED			STARTEDGESTS			
R-0h	R-0-0h			R/W-0h			

**Table 18-36. VCNTCFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	STOPEDESTS	R	0h	Stop Edge Status Bit 0: Stop edge has not occurred 1: Stop edge occurred  Note: This bit is set only after the trigger sequence is armed (upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL]) and STOPEDESTS occurs.  Note: This bit is reset by the occurrence of the trigger pulse selected through VCAPCTL[TRIGSEL] Reset type: SYSRSn
14-12	RESERVED	R-0	0h	Reserved
11-8	STOPEDESTS	R/W	0h	Counter Stop Edge Selection  Once the counter operation is armed, upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL] pulse - valley counter would stop counting upon the occurrence of chosen number of events through this bit field. Stop counting on occurrence of:  0000: Do not stop 0001 1st edge 0010: 2nd edge 0011: 3rd edge ... 1,1,1,1: 15th edge Reset type: SYSRSn
7	STARTEDGESTS	R	0h	Start Edge Status Bit 0: Start edge has not occurred 1: Start edge occurred  Note: This bit is set only after the trigger sequence is armed (upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL]) and STARTEDGESTS occurs.  Note: This bit is reset by the occurrence of the trigger pulse selected through VCAPCTL[TRIGSEL] Reset type: SYSRSn
6-4	RESERVED	R-0	0h	Reserved

**Table 18-36. VCNTCFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	STARTEDGE	R/W	0h	<p>Counter Start Edge Selection</p> <p>Once the counter operation is armed, upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL] pulse - valley counter would start counting upon the occurrence of chosen number of events thorough this bit field. Start counting on occurrence of</p> <p>0000: Do not start  0001: 1st edge  0010: 2nd edge  0011: 3rd edge  ...  1111: 15th edge</p> <p>Reset type: SYSRSn</p>

### 18.16.2.14 HRCNFG Register (Offset = 20h) [reset = 0h]

HRCNFG is shown in [Figure 18-107](#) and described in [Table 18-37](#).

Return to the [Summary Table](#).

HRPWM Configuration Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 18-107. HRCNFG Register**

15	14	13	12	11	10	9	8
RESERVED		RESERVED	HRLOADB	CTLMODEB	EDGMODEB		
		R-0-0h	R/W-0h	R/W-0h	R/W-0h		
7	6	5	4	3	2	1	0
SWAPAB	AUTOCONV	SELOUTB	HRLOAD	CTLMODE	EDGMODE		
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		

**Table 18-37. HRCNFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R/W	0h	Reserved
13	RESERVED	R-0	0h	Reserved
12-11	HRLOADB	R/W	0h	Shadow Mode Bit  Selects the time event that loads the CMPBHR shadow value into the active register.  00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Reserved Reset type: SYSRSn
10	CTLMODEB	R/W	0h	Control Mode Bits  Selects the register (CMP/TBPRD or TBPHS) that controls the MEP:  0: CMPBHR(8) or TBPRDHR(8) Register controls the edge position (i.e., this is duty or period control mode). (Default on Reset) 1: TBPHSHR(8) Register controls the edge position (i.e., this is phase control mode). Reset type: SYSRSn
9-8	EDGMODEB	R/W	0h	Edge Mode Bits  Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic:  00: HRPWM capability is disabled (default on reset) 01: MEP control of rising edge (CMPBHR) 10: MEP control of falling edge (CMPBHR) 11: MEP control of both edges (TBPHSHR or TBPRDHR) Reset type: SYSRSn
7	SWAPAB	R/W	0h	Swap ePWM A & B Output Signals  This bit enables the swapping of the A & B signal outputs. The selection is as follows:  0: ePWMxA and ePWMxB outputs are unchanged. 1: ePWMxA signal appears on ePWMxB output and ePWMxB signal appears on ePWMxA output. Reset type: SYSRSn

**Table 18-37. HRCNFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	AUTOCONV	R/W	0h	<p>Auto Convert Delay Line Value</p> <p>Selects whether the fractional duty cycle/period/phase in the CMPAHR/TBPRDHR/TBPHSHR register is automatically scaled by the MEP scale factor in the HRMSTEP register or manually scaled by calculations in application software. The SFO library function automatically updates the HRMSTEP register with the appropriate MEP scale factor.</p> <p>0: Automatic HRMSTEP scaling is disabled. 1: Automatic HRMSTEP scaling is enabled.</p> <p>If application software is manually scaling the fractional duty cycle, or phase (i.e. software sets <math>CMPAHR = (\text{fraction}(PWMduty * PWMperiod) * MEP \text{ Scale Factor}) &lt; 8 + 0x080</math> for duty cycle), then this mode must be disabled.</p> <p>Reset type: SYSRSn</p>
5	SELOUTB	R/W	0h	<p>EPWMxB Output Select Bit</p> <p>This bit selects which signal is output on the ePWMxB channel output.</p> <p>The inversion will take the high resolution mode into account and the inverted signal will contain any high resolution modification. The inversion takes place as the last step in modifying the ePWMxB signal.</p> <p>0: ePWMxB output is normal. 1: ePWMxB output is inverted version of ePWMxA signal.</p> <p>Reset type: SYSRSn</p>
4-3	HRLOAD	R/W	0h	<p>Shadow Mode Bit</p> <p>Selects the time event that loads the CMPAHR shadow value into the active register.</p> <p>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Reserved</p> <p>Reset type: SYSRSn</p>
2	CTLMODE	R/W	0h	<p>Control Mode Bits</p> <p>Selects the register (CMP/TBPRD or TBPHS) that controls the MEP:</p> <p>0: CMPAHR(8) or TBPRDHR(8) Register controls the edge position (i.e., this is duty or period control mode). (Default on Reset) 1: TBPHSHR(8) Register controls the edge position (i.e., this is phase control mode).</p> <p>Reset type: SYSRSn</p>
1-0	EDGMODE	R/W	0h	<p>Edge Mode Bits</p> <p>Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic:</p> <p>00: HRPWM capability is disabled (default on reset) 01: MEP control of rising edge (CMPAHR) 10: MEP control of falling edge (CMPAHR) 11: MEP control of both edges (TBPHSHR or TBPRDHR)</p> <p>Reset type: SYSRSn</p>

### 18.16.2.15 HRPWR Register (Offset = 21h) [reset = 0h]

HRPWR is shown in [Figure 18-108](#) and described in [Table 18-38](#).

Return to the [Summary Table](#).

HRPWM Power Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 18-108. HRPWR Register**

15	14	13	12	11	10	9	8
CALPWRON	RESERVED					RESERVED	
R/W-0h		R-0-0h					
7	6	5	4	3	2	1	0
RESERVED		RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	

**Table 18-38. HRPWR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	CALPWRON	R/W	0h	MEP Calibration Power Bits 0: Disables MEP calibration logic in the HRPWM and reduces power consumption. 1: Enables MEP calibration logic Reset type: SYSRSn
14-10	RESERVED	R-0	0h	Reserved
9-6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1-0	RESERVED	R/W	0h	Reserved

**18.16.2.16 HRMSTEP Register (Offset = 26h) [reset = 0h]**

HRMSTEP is shown in [Figure 18-109](#) and described in [Table 18-39](#).

Return to the [Summary Table](#).

**HRPWM MEP Step Register**

This register is only accessible on EPWM modules with HRPWM capabilities. Only 16 bit accesses are allowed for this register. Debugger access in 32 bit mode may display incorrect values.

**Figure 18-109. HRMSTEP Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
HRMSTEP							
R/W-0h							

**Table 18-39. HRMSTEP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-0	HRMSTEP	R/W	0h	<p>High Resolution MEP Step</p> <p>When auto-conversion is enabled (HRCNFG[AUTOCONV] = 1), This 8-bit field contains the MEP_ScaleFactor (number of MEP steps per coarse steps) used by the hardware to automatically convert the value in the CMPAHR, CMPBHR, DBFEDHR, DBREDHR, TBPHSHR, or TBPRDHR register to a scaled micro-edge delay on the high-resolution ePWM output. The value in this register is written by the SFO calibration software at the end of each calibration run.</p> <p>Reset type: SYSRSn</p>



**18.16.2.17 HRCNFG2 Register (Offset = 27h) [reset = 0h]**

HRCNFG2 is shown in [Figure 18-110](#) and described in [Table 18-40](#).

Return to the [Summary Table](#).

**HRPWM Configuration 2 Register**

This register is only accessible on ePWM modules with HRPWM capabilities. Only 16 bit accesses are allowed for this register. Debugger access in 32 bit mode may display incorrect values.

**Figure 18-110. HRCNFG2 Register**

15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED					
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		CTLMODEDBFED		CTLMODEDBRED		EDGMODEDB	
R-0-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 18-40. HRCNFG2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13-6	RESERVED	R-0	0h	Reserved
5-4	CTLMODEDBFED	R/W	0h	Shadow Mode Bit - selection should match DBCTL[LOADFEDMODE]  Selects the time event that loads the DBFEDHR shadow value into the active register.  00 Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01 Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10 Load on either CTR = Zero or CTR = PRD 11 Reserved Reset type: SYSRSn
3-2	CTLMODEDBRED	R/W	0h	Shadow Mode Bit - selection should match DBCTL[LOADREDMODE]  Selects the time event that loads the DBREDHR shadow value into the active register.  00 Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01 Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10 Load on either CTR = Zero or CTR = PRD 11 Reserved Reset type: SYSRSn
1-0	EDGMODEDB	R/W	0h	Edge Mode Bits  Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic:  00 HRPWM capability is disabled (default on reset) 01 MEP control of rising edge (DBREDHR) 10 MEP control of falling edge (DBFEDHR) 11 MEP control of both edges (rising edge of DBREDHR or falling edge of DBFEDHR ) Reset type: SYSRSn

**18.16.2.18 HRPCTL Register (Offset = 2Dh) [reset = 0h]**

HRPCTL is shown in [Figure 18-111](#) and described in [Table 18-41](#).

Return to the [Summary Table](#).

High Resolution Period Control Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 18-111. HRPCTL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	PWMSYNCSELX			RESERVED	TBPHSHRLOA DE	PWMSYNCSEL	HRPE
R-0-0h	R/W-0h				R/W-0h	R/W-0h	R/W-0h

**Table 18-41. HRPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6-4	PWMSYNCSELX	R/W	0h	Extended selection bits for EPWMSYNCPER  000: EPWMSYNCPER is defined by PWMSYNCSEL - > default condition (compatible with previous EPWM versions)  001: Reserved 010: Reserved 011: Reserved 100: CTR = CMPC, Count direction Up 101: CTR = CMPC, Count direction Down 110: CTR = CMPD, Count direction Up 111: CTR = CMPD, Count direction Down Reset type: SYSRSn
3	RESERVED	R/W	0h	Reserved
2	TBPHSHRLOADE	R/W	0h	TBPHSHR Load Enable  This bit allows you to synchronize ePWM modules with a high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital compare event. This allows for multiple ePWM modules operating at the same frequency to be phase aligned with high-resolution.  0: Disables synchronization of high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital compare event:  1: Synchronize the high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital comparator synchronization event. The phase is synchronized using the contents of the high-resolution phase TBPHSHR register. The TBCTL[PHSEN] bit which enables the loading of the TBCTR register with TBPHS register value on a SYNCIN or TBCTL[SWFSYNC] event works independently. However, users need to enable this bit also if they want to control phase in conjunction with the high-resolution period feature.  This bit and the TBCTL[PHSEN] bit must be set to 1 when high-resolution period is enabled for up-down count mode even if TBPHSHR = 0x0000. This bit does not need to be set when only high-resolution duty is enabled.  Reset type: SYSRSn

**Table 18-41. HRPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	PWMSYNCSEL	R/W	0h	<p>PWMSYNC Source Select Bit: This bit selects the source for the EPWMSYNCPER signal that goes to the CMPSS and GPDAC:</p> <p>0 CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)</p> <p>1 CTR = zero: Time-base counter equal to zero (TBCTR = 0x00)</p> <p>Reset type: SYSRSn</p>
0	HRPE	R/W	0h	<p>High Resolution Period Enable Bit</p> <p>0: High resolution period feature disabled. In this mode the ePWM behaves as a Type 0 ePWM.</p> <p>1: High resolution period enabled. In this mode the HRPWM module can control high-resolution of both the duty and frequency. When high-resolution period is enabled, TBCTL[CTRMODE] = 0,1 (down-count mode) is not supported.</p> <p>Reset type: SYSRSn</p>

**18.16.2.19 TRREM Register (Offset = 2Eh) [reset = 0h]**

TRREM is shown in [Figure 18-112](#) and described in [Table 18-42](#).

Return to the [Summary Table](#).

Translator High Resolution Remainder Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 18-112. TRREM Register**

15	14	13	12	11	10	9	8
RESERVED					TRREM		
R-0-0h					R/W-0h		
7	6	5	4	3	2	1	0
TRREM							
R/W-0h							

**Table 18-42. TRREM Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R-0	0h	Reserved
10-0	TRREM	R/W	0h	Translator Remainder Bits: This 11-bit value keeps track of the remainder portion of the translator algorithm calculations. This value keeps track of the remainder portion of the HRPWM hardware calculations. Notes: 1. The lower 8-bits of the TRREM register can be automatically initialized with the TBPHSHR value on a SYNCIN or TBCTL[SWFSYNC] event or DC event (if enabled). The user can also write a value with the CPU. 2. Priority of TRREM register updates: Sync (software or hardware) TBPHSHR copied to TRREM : Highest Priority HRPWM Hardware (updates TRREM register): Next priority CPU Write To TRREM Register: Lowest Priority 3. Bit 10 of TRREM register is not used in asymmetrical mode. This bit can be forced to zero. Reset type: SYSRSn

### 18.16.2.20 GLDCTL Register (Offset = 34h) [reset = 0h]

GLDCTL is shown in [Figure 18-113](#) and described in [Table 18-43](#).

Return to the [Summary Table](#).

Global PWM Load Control Register

**Figure 18-113. GLDCTL Register**

15	14	13	12	11	10	9	8
RESERVED			GLDCNT			GLDPRD	
R-0-0h			R-0h			R/W-0h	
7	6	5	4	3	2	1	0
GLDPRD	RESERVED	OSHTMODE	GLDMODE			GLD	
R/W-0h	R-0-0h	R/W-0h	R/W-0h			R/W-0h	

**Table 18-43. GLDCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R-0	0h	Reserved
12-10	GLDCNT	R	0h	Global Load Strobe Counter Register These bits indicate how many selected events have occurred: 000: No events 001: 1 event 010: 2 events 011: 3 events 100: 4 events 101: 5 events 110: 6 events 111: 7 events Reset type: SYSRSn
9-7	GLDPRD	R/W	0h	Global Load Strobe Period Select Register These bits select how many selected events need to occur before a load strobe is generated 000: Disable counter 001: Generate strobe on GLDCNT = 001 (1st event) 010: Generate strobe on GLDCNT = 010 (2nd event) 011: Generate strobe on GLDCNT = 011 (3rd event) 100: Generate strobe on GLDCNT = 011 (4th event) 101: Generate strobe on GLDCNT = 001 (5th event) 110: Generate strobe on GLDCNT = 010 (6th event) 111: Generate strobe on GLDCNT = 011 (7th event) Reset type: SYSRSn
6	RESERVED	R-0	0h	Reserved
5	OSHTMODE	R/W	0h	One Shot Load Mode Control Bit 0: One shot load mode is disabled and shadow to active loading happens continuously on all the chosen load strobes. 1: One shot mode is active. All load strobes are blocked until GLDCTL2[OSHTLD] is written with 1. Note: One Shot mode can only be used with global shadow to active load mode enabled (GLDCTL[GLD]=1) Reset type: SYSRSn

**Table 18-43. GLDCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-1	GLDMODE	R/W	0h	<p>Global Load Pulse selection for Shadow to Active Mode Reloads</p> <p>0000: Load on Counter = 0 (CNT_ZRO)</p> <p>0001: Load on Counter = Period (PRD_EQ)</p> <p>0010: Load on either Counter = 0, or Counter = Period</p> <p>0011: Load on SYNCEVT - this is logical OR of DCAEVT1.sync, DCBEVT1.sync, EPWMxSYNCl and TBCTL[SWFSYNC]</p> <p>0100: Load on SYNCEVT or CNT_ZRO</p> <p>0101: Load on SYNCEVT or PRD_EQ</p> <p>0110: Load on SYNCEVT or CNT_ZRO or PRD_EQ</p> <p>1000: Reserved</p> <p>...</p> <p>1110: Reserved</p> <p>1111: Load on GLDCTL[GLDFRCLD] write</p> <p>Reset type: SYSRSn</p>
0	GLD	R/W	0h	<p>Global Shadow to Active Load Event Control</p> <p>0: Shadow to active reload for all shadowed registers happens as per the individual reload control bits specified (Compatible with previous EPWM versions).</p> <p>1: When set, all the shadow to active reload events are defined by GLDMODE bits in GLDCTL register. All the shadow registers use same reload pulse from shadow to active reloading. Individual LOADMODE bits are ignored.</p> <p>Reset type: SYSRSn</p>

### 18.16.2.21 GLDCFG Register (Offset = 35h) [reset = 0h]

GLDCFG is shown in [Figure 18-114](#) and described in [Table 18-44](#).

Return to the [Summary Table](#).

Global PWM Load Config Register

**Figure 18-114. GLDCFG Register**

15	14	13	12	11	10	9	8
RESERVED					AQCSFRC	AQCTLB_AQC TLB2	AQCTLA_AQC TLA2
R-0-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DBCTL	DBFED_DBFE DHR	DBRED_DBRE DHR	CMPD	CMPC	CMPB_CMPBH R	CMPA_CMPAH R	TBPRD_TBPR DHR
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 18-44. GLDCFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R-0	0h	Reserved
10	AQCSFRC	R/W	0h	Global load event configuration for AQCSFRC  0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
9	AQCTLB_AQCTLB2	R/W	0h	Global load event configuration for AQCTLB_AQCTLB2  0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
8	AQCTLA_AQCTLA2	R/W	0h	Global load event configuration for AQCTLA_AQCTLA2  0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
7	DBCTL	R/W	0h	Global load event configuration for DBCTL  0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
6	DBFED_DBFEDHR	R/W	0h	Global load event configuration for DBFED_DBFEDHR  0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn

**Table 18-44. GLDCFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	DBRED_DBREDHR	R/W	0h	Global load event configuration for DBRED_DBREDHR 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
4	CMPD	R/W	0h	Global load event configuration for CMPD 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
3	CMPC	R/W	0h	Global load event configuration for CMPC 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
2	CMPB_CMPBHR	R/W	0h	Global load event configuration for CMPB_CMPBHR 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
1	CMPA_CMPAHR	R/W	0h	Global load event configuration for CMPA_CMPAHR 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
0	TBPRD_TBPRDHR	R/W	0h	Global load event configuration for TBPRD_TBPRDHR 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn



### 18.16.2.22 EPWMXLINK Register (Offset = 38h) [reset = X]

EPWMXLINK is shown in [Figure 18-115](#) and described in [Table 18-45](#).

Return to the [Summary Table](#).

#### EPWMx Link Register

This register controls which EPWMs are linked to other EPWM modules. The default reset value will vary for each module. The reset value will link each EPWM module to itself to prevent unintentional linking of modules.

**Figure 18-115. EPWMXLINK Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GLDCTL2LINK				RESERVED								CMPDLINK			
R/W-X				R-0-0h								R/W-X			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPCLINK				CMPBLINK				CMPALINK				TBPRDLINK			
R/W-X				R/W-X				R/W-X				R/W-X			

**Table 18-45. EPWMXLINK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	GLDCTL2LINK	R/W	X	<p>GLDCTL2 Link Bits</p> <p>Writes to the GLDCTL2 registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's GLDCTL2 registers.</p> <p>0000: ePWM1            0001: ePWM2            0010: ePWM3            0011: ePWM4            0100: ePWM5            0101: ePWM6            0110: ePWM7            0111: ePWM8            1000: ePWM9            1001: ePWM10            1010: ePWM11            1011: ePWM12            1100: Reserved            ...            1111: Reserved            Reset type: SYSRSn</p>
27-20	RESERVED	R-0	0h	Reserved

**Table 18-45. EPWMXLINK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-16	CMPDLINK	R/W	X	<p>CMPD Link Bits</p> <p>Writes to the CMPD registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPD registers.</p> <p>0000: ePWM1  0001: ePWM2  0010: ePWM3  0011: ePWM4  0100: ePWM5  0101: ePWM6  0110: ePWM7  0111: ePWM8  1000: ePWM9  1001: ePWM10  1010: ePWM11  1011: ePWM12  1100: Reserved  ...  1111: Reserved  Reset type: SYSRSn</p>
15-12	CMPLINK	R/W	X	<p>CMPC Link Bits</p> <p>Writes to the CMPC registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPC registers.</p> <p>0000: ePWM1  0001: ePWM2  0010: ePWM3  0011: ePWM4  0100: ePWM5  0101: ePWM6  0110: ePWM7  0111: ePWM8  1000: ePWM9  1001: ePWM10  1010: ePWM11  1011: ePWM12  1100: Reserved  ...  1111: Reserved  Reset type: SYSRSn</p>

**Table 18-45. EPWMXLINK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-8	CMPBLINK	R/W	X	CMPB_CMPBHR Link Bits  Writes to the CMPB_CMPBHR registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPB_CMPBHR registers.  0000: ePWM1 0001: ePWM2 0010: ePWM3 0011: ePWM4 0100: ePWM5 0101: ePWM6 0110: ePWM7 0111: ePWM8 1000: ePWM9 1001: ePWM10 1010: ePWM11 1011: ePWM12 1100: Reserved ... 1111: Reserved Reset type: SYSRSn
7-4	CMPALINK	R/W	X	CMPA_CMPAHR Link Bits  Writes to the CMPA_CMPAHR registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPA_CMPAHR registers.  0000: ePWM1 0001: ePWM2 0010: ePWM3 0011: ePWM4 0100: ePWM5 0101: ePWM6 0110: ePWM7 0111: ePWM8 1000: ePWM9 1001: ePWM10 1010: ePWM11 1011: ePWM12 1100: Reserved ... 1111: Reserved Reset type: SYSRSn

**Table 18-45. EPWMXLINK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	TBPRDLINK	R/W	X	<p>TBPRD_TBPRDHR Link Bits</p> <p>Writes to the TBPRD:TBPRDHR registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's TBPRD_TBPRDHR registers.</p> <p>0000: ePWM1  0001: ePWM2  0010: ePWM3  0011: ePWM4  0100: ePWM5  0101: ePWM6  0110: ePWM7  0111: ePWM8  1000: ePWM9  1001: ePWM10  1010: ePWM11  1011: ePWM12  1100: Reserved  ...  1111: Reserved</p> <p>Reset type: SYSRSn</p>

### 18.16.2.23 AQCTLA Register (Offset = 40h) [reset = 0h]

AQCTLA is shown in [Figure 18-116](#) and described in [Table 18-46](#).

Return to the [Summary Table](#).

Action Qualifier Control Register For Output A

**Figure 18-116. AQCTLA Register**

15	14	13	12	11	10	9	8
RESERVED				CBD		CBU	
R-0-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 18-46. AQCTLA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-10	CBD	R/W	0h	Action When TBCTR = CMPB on Down Count  Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.  00: Do nothing (action disabled) 01: Clear: force EPWMxA output low. 10: Set: force EPWMxA output high. 11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
9-8	CBU	R/W	0h	Action When TBCTR = CMPB on Up Count  Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.  00: Do nothing (action disabled) 01: Clear: force EPWMxA output low. 10: Set: force EPWMxA output high. 11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
7-6	CAD	R/W	0h	Action When TBCTR = CMPA on Down Count  Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.  00: Do nothing (action disabled) 01: Clear: force EPWMxA output low. 10: Set: force EPWMxA output high. 11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn

**Table 18-46. AQCTLA Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-4	CAU	R/W	0h	<p>Action When TBCTR = CMPA on Up Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)  01: Clear: force EPWMxA output low.  10: Set: force EPWMxA output high.  11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>
3-2	PRD	R/W	0h	<p>Action When TBCTR = TBPRD</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)  01: Clear: force EPWMxA output low.  10: Set: force EPWMxA output high.  11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>
1-0	ZRO	R/W	0h	<p>Action When TBCTR = 0</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)  01: Clear: force EPWMxA output low.  10: Set: force EPWMxA output high.  11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>

**18.16.2.24 AQCTLA2 Register (Offset = 41h) [reset = 0h]**

AQCTLA2 is shown in [Figure 18-117](#) and described in [Table 18-47](#).

Return to the [Summary Table](#).

Additional Action Qualifier Control Register For Output A

**Figure 18-117. AQCTLA2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
T2D		T2U		T1D		T1U	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 18-47. AQCTLA2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-6	T2D	R/W	0h	<p>Action when event occurs on T2 in DOWN-Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)</p> <p>01: Clear: force EPWMxA output low.</p> <p>10: Set: force EPWMxA output high.</p> <p>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>
5-4	T2U	R/W	0h	<p>Action when event occurs on T2 in UP-Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)</p> <p>01: Clear: force EPWMxA output low.</p> <p>10: Set: force EPWMxA output high.</p> <p>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>
3-2	T1D	R/W	0h	<p>Action when event occurs on T1 in DOWN-Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)</p> <p>01: Clear: force EPWMxA output low.</p> <p>10: Set: force EPWMxA output high.</p> <p>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>

**Table 18-47. AQCTLA2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	T1U	R/W	0h	<p>Action when event occurs on T1 in UP-Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)</p> <p>01: Clear: force EPWMxA output low.</p> <p>10: Set: force EPWMxA output high.</p> <p>11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>



### 18.16.2.25 AQCTLB Register (Offset = 42h) [reset = 0h]

AQCTLB is shown in [Figure 18-118](#) and described in [Table 18-48](#).

Return to the [Summary Table](#).

Action Qualifier Control Register For Output B

**Figure 18-118. AQCTLB Register**

15	14	13	12	11	10	9	8
RESERVED				CBD		CBU	
R-0-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 18-48. AQCTLB Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-10	CBD	R/W	0h	Action When TBCTR = CMPB on Down Count  Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.  00: Do nothing (action disabled) 01: Clear: force EPWMxB output low. 10: Set: force EPWMxB output high. 11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
9-8	CBU	R/W	0h	Action When TBCTR = CMPB on Up Count  Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.  00: Do nothing (action disabled) 01: Clear: force EPWMxB output low. 10: Set: force EPWMxB output high. 11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
7-6	CAD	R/W	0h	Action When TBCTR = CMPA on Down Count  Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.  00: Do nothing (action disabled) 01: Clear: force EPWMxB output low. 10: Set: force EPWMxB output high. 11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn

**Table 18-48. AQCTLB Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-4	CAU	R/W	0h	<p>Action When TBCTR = CMPA on Up Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)  01: Clear: force EPWMxB output low.  10: Set: force EPWMxB output high.  11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>
3-2	PRD	R/W	0h	<p>Action When TBCTR = TBPRD</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)  01: Clear: force EPWMxB output low.  10: Set: force EPWMxB output high.  11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>
1-0	ZRO	R/W	0h	<p>Action When TBCTR = 0</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)  01: Clear: force EPWMxB output low.  10: Set: force EPWMxB output high.  11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>

**18.16.2.26 AQCTLB2 Register (Offset = 43h) [reset = 0h]**

AQCTLB2 is shown in [Figure 18-119](#) and described in [Table 18-49](#).

Return to the [Summary Table](#).

Additional Action Qualifier Control Register For Output B

**Figure 18-119. AQCTLB2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
T2D		T2U		T1D		T1U	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 18-49. AQCTLB2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-6	T2D	R/W	0h	<p>Action when event occurs on T2 in DOWN-Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)</p> <p>01: Clear: force EPWMxB output low.</p> <p>10: Set: force EPWMxB output high.</p> <p>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>
5-4	T2U	R/W	0h	<p>Action when event occurs on T2 in UP-Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)</p> <p>01: Clear: force EPWMxB output low.</p> <p>10: Set: force EPWMxB output high.</p> <p>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>
3-2	T1D	R/W	0h	<p>Action when event occurs on T1 in DOWN-Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)</p> <p>01: Clear: force EPWMxB output low.</p> <p>10: Set: force EPWMxB output high.</p> <p>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>

**Table 18-49. AQCTLB2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	T1U	R/W	0h	<p>Action when event occurs on T1 in UP-Count</p> <p>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.</p> <p>00: Do nothing (action disabled)</p> <p>01: Clear: force EPWMxB output low.</p> <p>10: Set: force EPWMxB output high.</p> <p>11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.</p> <p>Reset type: SYSRSn</p>

**18.16.2.27 AQSFR Register (Offset = 47h) [reset = 0h]**

AQSFR is shown in [Figure 18-120](#) and described in [Table 18-50](#).

Return to the [Summary Table](#).

Action Qualifier Software Force Register

**Figure 18-120. AQSFR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RLDCSF		OTSFB		ACTSFB		OTSFA	
R/W-0h		R-0/W1S-0h		R/W-0h		R/W-0h	

**Table 18-50. AQSFR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-6	RLDCSF	R/W	0h	AQCSFR Active Register Reload From Shadow Options 00: Load on event counter equals zero 01: Load on event counter equals period 10: Load on event counter equals zero or counter equals period 11: Load immediately (the active register is directly accessed by the CPU and is not loaded from the shadow register). Reset type: SYSRSn
5	OTSFB	R-0/W1S	0h	One-Time Software Forced Event on Output B 0: Writing a 0 (zero) has no effect. Always reads back a 0. This bit is auto cleared once a write to this register is complete (i.e., a forced event is initiated.). This is a one-shot forced event. It can be overridden by another subsequent event on output B. 1: Initiates a single software forced event Reset type: SYSRSn
4-3	ACTSFB	R/W	0h	Action When One-Time Software Force B is Invoked 00: Does nothing (action disabled) 01: Clear (low) 10: Set (high) 11: Toggle (Low -> High, High -> Low) Note: This action is not qualified by counter direction (CNT_dir) Reset type: SYSRSn
2	OTSFA	R-0/W1S	0h	One-Time Software Forced Event on Output A 0: Writing a 0 (zero) has no effect. Always reads back a 0. This bit is auto cleared once a write to this register is complete (i.e., a forced event is initiated). This is a one-shot forced event. It can be overridden by another subsequent event on output A. 1: Initiates a single software forced event Reset type: SYSRSn

**Table 18-50. AQSFR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	ACTSFA	R/W	0h	Action When One-Time Software Force A Is Invoked 00: Does nothing (action disabled) 01: Clear (low) 10: Set (high) 11: Toggle (Low -> High, High -> Low) Note: This action is not qualified by counter direction (CNT_dir) Reset type: SYSRSn

**18.16.2.28 AQCSFRC Register (Offset = 49h) [reset = 0h]**

AQCSFRC is shown in [Figure 18-121](#) and described in [Table 18-51](#).

Return to the [Summary Table](#).

Action Qualifier Continuous S/W Force Register

**Figure 18-121. AQCSFRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				CSFB		CSFA	
R-0-0h				R/W-0h		R/W-0h	

**Table 18-51. AQCSFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	CSFB	R/W	0h	<p>Continuous Software Force on Output B</p> <p>In immediate mode, a continuous force takes effect on the next TBCLK edge. In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register. To configure shadow mode, use AQSFRC[RLDCSF].</p> <p>00: Software forcing is disabled and has no effect            01: Forces a continuous low on output B            10: Forces a continuous high on output B            11: Software forcing is disabled and has no effect</p> <p>Reset type: SYSRSn</p>
1-0	CSFA	R/W	0h	<p>Continuous Software Force on Output A</p> <p>In immediate mode, a continuous force takes effect on the next TBCLK edge. In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register.</p> <p>00: Software forcing is disabled and has no effect            01: Forces a continuous low on output A            10: Forces a continuous high on output A            11: Software forcing is disabled and has no effect</p> <p>Reset type: SYSRSn</p>

**18.16.2.29 DBREDHR Register (Offset = 50h) [reset = 0h]**

DBREDHR is shown in [Figure 18-122](#) and described in [Table 18-52](#).

Return to the [Summary Table](#).

Dead-Band Generator Rising Edge Delay High Resolution Mirror Register

**Figure 18-122. DBREDHR Register**

15	14	13	12	11	10	9	8
DBREDHR							RESERVED
R/W-0h							R-0h
7	6	5	4	3	2	1	0
RESERVED							RESERVED
							R-0h

**Table 18-52. DBREDHR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	DBREDHR	R/W	0h	Dead Band Rising Edge Delay High Resolution Bits Reset type: SYSRSn
8	RESERVED	R	0h	Reserved
7-1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved



**18.16.2.30 DBRED Register (Offset = 51h) [reset = 0h]**

DBRED is shown in [Figure 18-123](#) and described in [Table 18-53](#).

Return to the [Summary Table](#).

Dead-Band Generator Rising Edge Delay High Resolution Mirror Register

**Figure 18-123. DBRED Register**

15	14	13	12	11	10	9	8
RESERVED				DBRED			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
DBRED							
R/W-0h							

**Table 18-53. DBRED Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0h	Reserved
13-0	DBRED	R/W	0h	Rising edge delay value Reset type: SYSRSn

**18.16.2.31 DBFEDHR Register (Offset = 52h) [reset = 0h]**

DBFEDHR is shown in [Figure 18-124](#) and described in [Table 18-54](#).

Return to the [Summary Table](#).

Dead-Band Generator Falling Edge Delay High Resolution Register

**Figure 18-124. DBFEDHR Register**

15	14	13	12	11	10	9	8
DBFEDHR							RESERVED
R/W-0h							R-0h
7	6	5	4	3	2	1	0
RESERVED							RESERVED
							R-0h

**Table 18-54. DBFEDHR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	DBFEDHR	R/W	0h	Dead Band Falling Edge Delay High Resolution Bits Reset type: SYSRSn
8	RESERVED	R	0h	Reserved
7-1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

**18.16.2.32 DBFED Register (Offset = 53h) [reset = 0h]**

DBFED is shown in [Figure 18-125](#) and described in [Table 18-55](#).

Return to the [Summary Table](#).

Dead-Band Generator Falling Edge Delay Count Register

**Figure 18-125. DBFED Register**

15	14	13	12	11	10	9	8
RESERVED				DBFED			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
DBFED							
R/W-0h							

**Table 18-55. DBFED Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0h	Reserved
13-0	DBFED	R/W	0h	Falling Edge Delay Count 14-bit counter Reset type: SYSRSn

**18.16.2.33 TBPHS Register (Offset = 60h) [reset = 0h]**

TBPHS is shown in [Figure 18-126](#) and described in [Table 18-56](#).

Return to the [Summary Table](#).

Time Base Phase High

**Figure 18-126. TBPHS 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
TBPHS																TBPHSHR															
R/W-0h																R/W-0h															

**Table 18-56. TBPHS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	TBPHS	R/W	0h	Phase Offset Register  These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal.  - If TBCTL[PHSEN] = 0, then the synchronization event is ignored and the time-base counter is not loaded with the phase. - If TBCTL[PHSEN] = 1, then the time-base counter (TBCTR) will be loaded with the phase (TBPHS) when a synchronization event occurs. The synchronization event can be initiated by the input synchronization signal (EPWMxSYNCl) or by a software forced synchronization. Reset type: SYSRSn
15-0	TBPHSHR	R/W	0h	Phase Offset (High Resolution) Register. TBPHSHR must not be used. Instead TRREM (translator remainder register) must be used to mimic the functionality of TBPHSHR. Reset type: SYSRSn

**18.16.2.34 TBPRDHR Register (Offset = 62h) [reset = 0h]**

TBPRDHR is shown in [Figure 18-127](#) and described in [Table 18-57](#).

Return to the [Summary Table](#).

Time Base Period High Resolution Register

**Figure 18-127. TBPRDHR Register**

15	14	13	12	11	10	9	8
TBPRDHR							
R/W-0h							
7	6	5	4	3	2	1	0
TBPRDHR							
R/W-0h							

**Table 18-57. TBPRDHR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TBPRDHR	R/W	0h	<p>Period High Resolution Bits</p> <p>The upper 8-bits contain the high-resolution portion of the period value. The TBPRDHR register is not affected by the TBCTL[PRDL] bit. Reads from this register always reflect the shadow register. Likewise writes are also to the shadow register. The TBPRDHR register is only used when the high resolution period feature is enabled. This register is only available with ePWM modules which support high-resolution period control.</p> <p>Reset type: SYSRSn</p>

**18.16.2.35 TBPRD Register (Offset = 63h) [reset = 0h]**

TBPRD is shown in [Figure 18-128](#) and described in [Table 18-58](#).

Return to the [Summary Table](#).

Time Base Period Register

**Figure 18-128. TBPRD Register**

15	14	13	12	11	10	9	8
TBPRD							
R/W-0h							
7	6	5	4	3	2	1	0
TBPRD							
R/W-0h							

**Table 18-58. TBPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TBPRD	R/W	0h	<p>Time Base Period Register</p> <p>These bits determine the period of the time-base counter. This sets the PWM frequency. Shadowing of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>- If TBCTL[PRDLD] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the active register will be loaded from the shadow register when the time-base counter equals zero.</li> <li>- If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>- The active and shadow registers share the same memory map address.</li> </ul> <p>Reset type: SYSRSn</p>

### 18.16.2.36 CMPA Register (Offset = 6Ah) [reset = 0h]

CMPA is shown in [Figure 18-129](#) and described in [Table 18-59](#).

Return to the [Summary Table](#).

Counter Compare A Register

**Figure 18-129. CMPA 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
CMPA																CMPAHR															
R/W-0h																R/W-0h															

**Table 18-59. CMPA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	CMPA	R/W	0h	<p>Compare A Register</p> <p>The value in the active CMPA register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare A" event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include:</p> <ul style="list-style-type: none"> <li>- Do nothing</li> <li>- Clear: Pull the EPWMxA and/or EPWMxB signal low</li> <li>- Set: Pull the EPWMxA and/or EPWMxB signal high</li> <li>- Toggle the EPWMxA and/or EPWMxB signal</li> </ul> <p>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWAMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>- If CMPCTL[SHDWAMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register.</li> <li>- Before a write, the CMPCTL[SHDWAFULL] bit can be read to determine if the shadow register is currently full.</li> <li>- If CMPCTL[SHDWAMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>- In either mode, the active and shadow registers share the same memory map address.</li> </ul> <p>Reset type: SYSRSn</p>
15-0	CMPAHR	R/W	0h	<p>Compare A HRPWM Extension Register</p> <p>These 8-bits contain the high-resolution portion (least significant 8-bits) of the counter-compare A value. CMPA:CMPAHR can be accessed in a single 32-bit read/write. Shadowing is enabled and disabled by the CMPCTL[SHDWAMODE] bit as described for the CMPA register.</p> <p>The lower 8 bits in this register are ignored</p> <p>Reset type: SYSRSn</p>

**18.16.2.37 CMPB Register (Offset = 6Ch) [reset = 0h]**

CMPB is shown in [Figure 18-130](#) and described in [Table 18-60](#).

Return to the [Summary Table](#).

Compare B Register

**Figure 18-130. CMPB 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
CMPB																CMPBHR															
R/W-0h																R/W-0h															

**Table 18-60. CMPB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	CMPB	R/W	0h	<p>Compare B Register</p> <p>The value in the active CMPB register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare B" event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include:</p> <ul style="list-style-type: none"> <li>- Do nothing</li> <li>- Clear: Pull the EPWMxA and/or EPWMxB signal low</li> <li>- Set: Pull the EPWMxA and/or EPWMxB signal high</li> <li>- Toggle the EPWMxA and/or EPWMxB signal</li> </ul> <p>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWBMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>- If CMPCTL[SHDWBMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADBMODE] bit field determines which event will load the active register from the shadow register.</li> <li>- Before a write, the CMPCTL[SHDWBFULL] bit can be read to determine if the shadow register is currently full.</li> <li>- If CMPCTL[SHDWBMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>- In either mode, the active and shadow registers share the same memory map address.</li> </ul> <p>Reset type: SYSRSn</p>
15-0	CMPBHR	R/W	0h	<p>Compare B High Resolution Bits</p> <p>The lower 8 bits in this register are ignored</p> <p>Reset type: SYSRSn</p>



### 18.16.2.38 CMPC Register (Offset = 6Fh) [reset = 0h]

CMPC is shown in [Figure 18-131](#) and described in [Table 18-61](#).

Return to the [Summary Table](#).

Counter Compare C Register

LINK feature access should always be 16-bit

**Figure 18-131. CMPC Register**

15	14	13	12	11	10	9	8
CMPC							
R/W-0h							
7	6	5	4	3	2	1	0
CMPC							
R/W-0h							

**Table 18-61. CMPC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	CMPC	R/W	0h	<p>Compare C Register</p> <p>The value in the active CMPC register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare C" event.</p> <p>Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWCMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>- If CMPCTL2[SHDWCMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL2[LOADCMODE] bit field determines which event will load the active register from the shadow register:</li> <li>- If CMPCTL2[SHDWCMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register that is, the register actively controlling the hardware.</li> <li>- In either mode, the active and shadow registers share the same memory map address.</li> </ul> <p>Reset type: SYSRSn</p>

**18.16.2.39 CMPD Register (Offset = 71h) [reset = 0h]**

CMPD is shown in [Figure 18-132](#) and described in [Table 18-62](#).

Return to the [Summary Table](#).

Counter Compare D Register

LINK feature access should always be 16-bit

**Figure 18-132. CMPD Register**

15	14	13	12	11	10	9	8
CMPD							
R/W-0h							
7	6	5	4	3	2	1	0
CMPD							
R/W-0h							

**Table 18-62. CMPD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	CMPD	R/W	0h	<p>Compare D Register</p> <p>The value in the active CMPD register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare D" event.</p> <p>Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWDMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>- If CMPCTL2[SHDWDMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL2[LOADDMODE] bit field determines which event will load the active register from the shadow register:</li> <li>- If CMPCTL2[SHDWDMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register that is, the register actively controlling the hardware.</li> <li>- In either mode, the active and shadow registers share the same memory map address.</li> </ul> <p>Reset type: SYSRSn</p>

**18.16.2.40 GLDCTL2 Register (Offset = 74h) [reset = 0h]**

GLDCTL2 is shown in [Figure 18-133](#) and described in [Table 18-63](#).

Return to the [Summary Table](#).

Global PWM Load Control Register 2

**Figure 18-133. GLDCTL2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						GFRCLD	OSHTLD
R-0-0h						R-0/W1S-0h	R-0/W1S-0h

**Table 18-63. GLDCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R-0	0h	Reserved
1	GFRCLD	R-0/W1S	0h	Force Load Event in One Shot Mode  0: Writing of 0 will be ignored. Always reads back a 0. 1: Force one load event at the input of the event pre-scale counter as shown in the diagram below. This bit is intended to be used for testing and/or software force loading of the events in global load mode.  Reset type: SYSRSn
0	OSHTLD	R-0/W1S	0h	Enable Reload Event in One Shot Mode  0: Writing of 0 will be ignored. Always reads back a 0. 1: Turns the one shot latch condition ON. Upon occurrence of a chosen load strobe, one shadow to active reload occurs and the latch will be cleared. Hence writing 1 to this bit would allow one load strobe event to pass through and block further strobe events.  Reset type: SYSRSn

**18.16.2.41 SWVDELVAL Register (Offset = 77h) [reset = 0h]**

SWVDELVAL is shown in [Figure 18-134](#) and described in [Table 18-64](#).

Return to the [Summary Table](#).

Software Valley Mode Delay Register

**Figure 18-134. SWVDELVAL Register**

15	14	13	12	11	10	9	8
SWVDELVAL							
R-0h							
7	6	5	4	3	2	1	0
SWVDELVAL							
R-0h							

**Table 18-64. SWVDELVAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SWVDELVAL	R	0h	<p>Software Valley Delay Value Register</p> <p>This register can be optionally used define offset value for the hardware calculated delay HWDELAYVAL as defined in VCAPCTL[VDELAYDIV] bits.</p> <p>Reset type: SYSRSn</p>

### 18.16.2.42 TZSEL Register (Offset = 80h) [reset = 0h]

TZSEL is shown in [Figure 18-135](#) and described in [Table 18-65](#).

Return to the [Summary Table](#).

Trip Zone Select Register

**Figure 18-135. TZSEL Register**

15		14		13		12		11		10		9		8	
DCBEVT1	DCAEVT1	OSHT6	OSHT5	OSHT4	OSHT3	OSHT2	OSHT1								
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	
DCBEVT2	DCAEVT2	CBC6	CBC5	CBC4	CBC3	CBC2	CBC1								
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 18-65. TZSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Select 0: Disable DCBEVT1 as one-shot-trip source for this ePWM module. 1: Enable DCBEVT1 as one-shot-trip source for this ePWM module. Reset type: SYSRSn
14	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Select 0: Disable DCAEVT1 as one-shot-trip source for this ePWM module. 1: Enable DCAEVT1 as one-shot-trip source for this ePWM module. Reset type: SYSRSn
13	OSHT6	R/W	0h	Trip-zone 6 (TZ6) Select 0: Disable TZ6 as a one-shot trip source for this ePWM module 1: Enable TZ6 as a one-shot trip source for this ePWM module Reset type: SYSRSn
12	OSHT5	R/W	0h	Trip-zone 5 (TZ5) Select 0: Disable TZ5 as a one-shot trip source for this ePWM module 1: Enable TZ5 as a one-shot trip source for this ePWM module Reset type: SYSRSn
11	OSHT4	R/W	0h	Trip-zone 4 (TZ4) Select 0: Disable TZ4 as a one-shot trip source for this ePWM module 1: Enable TZ4 as a one-shot trip source for this ePWM module Reset type: SYSRSn
10	OSHT3	R/W	0h	Trip-zone 3 (TZ3) Select 0: Disable TZ3 as a one-shot trip source for this ePWM module 1: Enable TZ3 as a one-shot trip source for this ePWM module Reset type: SYSRSn
9	OSHT2	R/W	0h	Trip-zone 2 (TZ2) Select 0: Disable TZ2 as a one-shot trip source for this ePWM module 1: Enable TZ2 as a one-shot trip source for this ePWM module Reset type: SYSRSn
8	OSHT1	R/W	0h	Trip-zone 1 (TZ1) Select 0: Disable TZ1 as a one-shot trip source for this ePWM module 1: Enable TZ1 as a one-shot trip source for this ePWM module Reset type: SYSRSn

**Table 18-65. TZSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Select 0: Disable DCBEVT2 as a CBC trip source for this ePWM module 1: Enable DCBEVT2 as a CBC trip source for this ePWM module Reset type: SYSRSn
6	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Select 0: Disable DCAEVT2 as a CBC trip source for this ePWM module 1: Enable DCAEVT2 as a CBC trip source for this ePWM module Reset type: SYSRSn
5	CBC6	R/W	0h	Trip-zone 6 (TZ6) Select 0: Disable TZ6 as a CBC trip source for this ePWM module 1: Enable TZ6 as a CBC trip source for this ePWM module Reset type: SYSRSn
4	CBC5	R/W	0h	Trip-zone 5 (TZ5) Select 0: Disable TZ5 as a CBC trip source for this ePWM module 1: Enable TZ5 as a CBC trip source for this ePWM module Reset type: SYSRSn
3	CBC4	R/W	0h	Trip-zone 4 (TZ4) Select 0: Disable TZ4 as a CBC trip source for this ePWM module 1: Enable TZ4 as a CBC trip source for this ePWM module Reset type: SYSRSn
2	CBC3	R/W	0h	Trip-zone 3 (TZ3) Select 0: Disable TZ3 as a CBC trip source for this ePWM module 1: Enable TZ3 as a CBC trip source for this ePWM module Reset type: SYSRSn
1	CBC2	R/W	0h	Trip-zone 2 (TZ2) Select 0: Disable TZ2 as a CBC trip source for this ePWM module 1: Enable TZ2 as a CBC trip source for this ePWM module Reset type: SYSRSn
0	CBC1	R/W	0h	Trip-zone 1 (TZ1) Select 0: Disable TZ1 as a CBC trip source for this ePWM module 1: Enable TZ1 as a CBC trip source for this ePWM module Reset type: SYSRSn

**18.16.2.43 TZDCSEL Register (Offset = 82h) [reset = 0h]**

 TZDCSEL is shown in [Figure 18-136](#) and described in [Table 18-66](#).

 Return to the [Summary Table](#).

Trip Zone Digital Comparator Select Register

**Figure 18-136. TZDCSEL Register**

15	14	13	12	11	10	9	8
RESERVED				DCBEVT2		DCBEVT1	
R-0-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
DCBEVT1		DCAEVT2			DCAEVT1		
R/W-0h		R/W-0h			R/W-0h		

**Table 18-66. TZDCSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-9	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Selection  000: Event disabled 001: DCBH = low, DCBL = don't care 010: DCBH = high, DCBL = don't care 011: DCBL = low, DCBH = don't care 100: DCBL = high, DCBH = don't care 101: DCBL = high, DCBH = low 110: Reserved 111: Reserved Reset type: SYSRSn
8-6	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Selection  000: Event disabled 001: DCBH = low, DCBL = don't care 010: DCBH = high, DCBL = don't care 011: DCBL = low, DCBH = don't care 100: DCBL = high, DCBH = don't care 101: DCBL = high, DCBH = low 110: Reserved 111: Reserved Reset type: SYSRSn
5-3	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Selection  000: Event disabled 001: DCAH = low, DCAL = don't care 010: DCAH = high, DCAL = don't care 011: DCAL = low, DCAH = don't care 100: DCAL = high, DCAH = don't care 101: DCAL = high, DCAH = low 110: Reserved 111: Reserved Reset type: SYSRSn

**Table 18-66. TZDCSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2-0	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Selection  000: Event disabled 001: DCAH = low, DCAL = don't care 010: DCAH = high, DCAL = don't care 011: DCAL = low, DCAH = don't care 100: DCAL = high, DCAH = don't care 101: DCAL = high, DCAH = low 110: Reserved 111: Reserved Reset type: SYSRSn



### 18.16.2.44 TZCTL Register (Offset = 84h) [reset = 0h]

TZCTL is shown in [Figure 18-137](#) and described in [Table 18-67](#).

Return to the [Summary Table](#).

Trip Zone Control Register

**Figure 18-137. TZCTL Register**

15	14	13	12	11	10	9	8
RESERVED				DCBEVT2		DCBEVT1	
R-0-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
DCAEVT2		DCAEVT1		TZB		TZA	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 18-67. TZCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-10	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Action On EPWMxB 00: High-impedance (EPWMxB = High-impedance state) 01: Force EPWMxB to a high state. 10: Force EPWMxB to a low state. 11: Do Nothing, trip action is disabled Reset type: SYSRSn
9-8	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Action On EPWMxB 00: High-impedance (EPWMxB = High-impedance state) 01: Force EPWMxB to a high state. 10: Force EPWMxB to a low state. 11: Do Nothing, trip action is disabled Reset type: SYSRSn
7-6	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Action On EPWMxA 00: High-impedance (EPWMxA = High-impedance state) 01: Force EPWMxA to a high state. 10: Force EPWMxA to a low state. 11: Do Nothing, trip action is disabled Reset type: SYSRSn
5-4	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Action On EPWMxA 00: High-impedance (EPWMxA = High-impedance state) 01: Force EPWMxA to a high state. 10: Force EPWMxA to a low state. 11: Do Nothing, trip action is disabled Reset type: SYSRSn
3-2	TZB	R/W	0h	TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxB When a trip event occurs the following action is taken on output EPWMxB. Which trip-zone pins can cause an event is defined in the TZSEL register. 00: High-impedance (EPWMxB = High-impedance state) 01: Force EPWMxB to a high state 10: Force EPWMxB to a low state 11: Do nothing, no action is taken on EPWMxB. Reset type: SYSRSn

**Table 18-67. TZCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	TZA	R/W	0h	<p>TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxA</p> <p>When a trip event occurs the following action is taken on output EPWMxA. Which trip-zone pins can cause an event is defined in the TZSEL register.</p> <p>00: High-impedance (EPWMxA = High-impedance state)            01: Force EPWMxA to a high state            10: Force EPWMxA to a low state            11: Do nothing, no action is taken on EPWMxA.</p> <p>Reset type: SYSRSn</p>

**18.16.2.45 TZCTL2 Register (Offset = 85h) [reset = 0h]**

TZCTL2 is shown in [Figure 18-138](#) and described in [Table 18-68](#).

Return to the [Summary Table](#).

Additional Trip Zone Control Register

**Figure 18-138. TZCTL2 Register**

15	14	13	12	11	10	9	8
ETZE	RESERVED			TZBD		TZBU	
R/W-0h	R-0-0h			R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
TZBU		TZAD			TZAU		
R/W-0h		R/W-0h			R/W-0h		

**Table 18-68. TZCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	ETZE	R/W	0h	TZCTL2 Enable  0: Use trip action from TZCTL (legacy EPWM compatibility) 1: Use trip action defined in TZCTL2, TZCTLDCB and TZCTLDCB. Settings in TZCTL are ignored Reset type: SYSRSn
14-12	RESERVED	R-0	0h	Reserved
11-9	TZBD	R/W	0h	TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxB while Count direction is DOWN  000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
8-6	TZBU	R/W	0h	TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxB while Count direction is UP  000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn

**Table 18-68. TZCTL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-3	TZAD	R/W	0h	TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxA while Count direction is DOWN  000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
2-0	TZAU	R/W	0h	TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxA while Count direction is UP  000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn

### 18.16.2.46 TZCTLDCA Register (Offset = 86h) [reset = 0h]

TZCTLDCA is shown in [Figure 18-139](#) and described in [Table 18-69](#).

Return to the [Summary Table](#).

Trip Zone Control Register Digital Compare A

**Figure 18-139. TZCTLDCA Register**

15	14	13	12	11	10	9	8
RESERVED				DCAEVT2D			DCAEVT2U
R-0-0h				R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
DCAEVT2U		DCAEVT1D			DCAEVT1U		
R/W-0h		R/W-0h			R/W-0h		

**Table 18-69. TZCTLDCA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-9	DCAEVT2D	R/W	0h	Digital Compare Output A Event 2 Action On EPWMxA while Count direction is DOWN  000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
8-6	DCAEVT2U	R/W	0h	Digital Compare Output A Event 2 Action On EPWMxA while Count direction is UP  000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
5-3	DCAEVT1D	R/W	0h	Digital Compare Output A Event 1 Action On EPWMxA while Count direction is DOWN  000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn

**Table 18-69. TZCTLDCA Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2-0	DCAEVT1U	R/W	0h	Digital Compare Output A Event 1 Action On EPWMxA while Count direction is UP  000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn

**18.16.2.47 TZCTLDCB Register (Offset = 87h) [reset = 0h]**

TZCTLDCB is shown in [Figure 18-140](#) and described in [Table 18-70](#).

Return to the [Summary Table](#).

Trip Zone Control Register Digital Compare B

**Figure 18-140. TZCTLDCB Register**

15	14	13	12	11	10	9	8
RESERVED				DCBEVT2D		DCBEVT2U	
R-0-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
DCBEVT2U		DCBEVT1D			DCBEVT1U		
R/W-0h		R/W-0h			R/W-0h		

**Table 18-70. TZCTLDCB Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-9	DCBEVT2D	R/W	0h	Digital Compare Output A Event 2 Action On EPWMxB while Count direction is DOWN  000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
8-6	DCBEVT2U	R/W	0h	Digital Compare Output A Event 2 Action On EPWMxB while Count direction is UP  000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
5-3	DCBEVT1D	R/W	0h	Digital Compare Output A Event 1 Action On EPWMxB while Count direction is DOWN  000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn

**Table 18-70. TZCTLDCB Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2-0	DCBEVT1U	R/W	0h	Digital Compare Output A Event 1 Action On EPWMxB while Count direction is UP  000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn



### 18.16.2.48 TZEINT Register (Offset = 8Dh) [reset = 0h]

TZEINT is shown in [Figure 18-141](#) and described in [Table 18-71](#).

Return to the [Summary Table](#).

Trip Zone Enable Interrupt Register

**Figure 18-141. TZEINT Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	RESERVED
R-0-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0-0h

**Table 18-71. TZEINT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Interrupt Enable 0: Disabled 1: Enabled Reset type: SYSRSn
5	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Interrupt Enable 0: Disabled 1: Enabled Reset type: SYSRSn
4	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Interrupt Enable 0: Disabled 1: Enabled Reset type: SYSRSn
3	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Interrupt Enable 0: Disabled 1: Enabled Reset type: SYSRSn
2	OST	R/W	0h	Trip-zone One-Shot Interrupt Enable 0: Disable one-shot interrupt generation 1: Enable Interrupt generation a one-shot trip event will cause a EPWMx_TZINT PIE interrupt. Reset type: SYSRSn
1	CBC	R/W	0h	Trip-zone Cycle-by-Cycle Interrupt Enable 0: Disable cycle-by-cycle interrupt generation. 1: Enable interrupt generation a cycle-by-cycle trip event will cause an EPWMx_TZINT PIE interrupt. Reset type: SYSRSn
0	RESERVED	R-0	0h	Reserved

**18.16.2.49 TZFLG Register (Offset = 93h) [reset = 0h]**

TZFLG is shown in [Figure 18-142](#) and described in [Table 18-72](#).

Return to the [Summary Table](#).

Trip Zone Flag Register

**Figure 18-142. TZFLG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	INT
R-0-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 18-72. TZFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6	DCBEVT2	R	0h	Latched Status Flag for Digital Compare Output B Event 2 0: Indicates no trip event has occurred on DCBEVT2 1: Indicates a trip event has occurred for the event defined for DCBEVT2 Reset type: SYSRSn
5	DCBEVT1	R	0h	Latched Status Flag for Digital Compare Output B Event 1 0: Indicates no trip event has occurred on DCBEVT1 1: Indicates a trip event has occurred for the event defined for DCBEVT1 Reset type: SYSRSn
4	DCAEVT2	R	0h	Latched Status Flag for Digital Compare Output A Event 2 0: Indicates no trip event has occurred on DCAEVT2 1: Indicates a trip event has occurred for the event defined for DCAEVT2 Reset type: SYSRSn
3	DCAEVT1	R	0h	Latched Status Flag for Digital Compare Output A Event 1 0: Indicates no trip event has occurred on DCAEVT1 1: Indicates a trip event has occurred for the event defined for DCAEVT1 Reset type: SYSRSn
2	OST	R	0h	Latched Status Flag for A One-Shot Trip Event 0: No one-shot trip event has occurred. 1: Indicates a trip event has occurred on a pin selected as a one-shot trip source.  This bit is cleared by writing the appropriate value to the TZCLR register. Reset type: SYSRSn

**Table 18-72. TZFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	CBC	R	0h	<p>Latched Status Flag for Cycle-By-Cycle Trip Event</p> <p>0: No cycle-by-cycle trip event has occurred. 1: Indicates a trip event has occurred on a signal selected as a cycle-by-cycle trip source. The</p> <p>TZFLG[CBC] bit will remain set until it is manually cleared by the user. If the cycle-by-cycle trip event is still present when the CBC bit is cleared, then CBC will be immediately set again. The specified condition on the signal is automatically cleared when the ePWM time-base counter reaches zero (TBCTR = 0x00) if the trip condition is no longer present. The condition on the signal is only cleared when the TBCTR = 0x00 no matter where in the cycle the CBC flag is cleared.</p> <p>This bit is cleared by writing the appropriate value to the TZCLR register. Reset type: SYSRSn</p>
0	INT	R	0h	<p>Latched Trip Interrupt Status Flag</p> <p>0: Indicates no interrupt has been generated. 1: Indicates an EPWMx_TZINT PIE interrupt was generated because of a trip condition.</p> <p>No further EPWMx_TZINT PIE interrupts will be generated until this flag is cleared. If the interrupt flag is cleared when either CBC or OST is set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts. This bit is cleared by writing the appropriate value to the TZCLR register. Reset type: SYSRSn</p>

**18.16.2.50 TZCBCFLG Register (Offset = 94h) [reset = 0h]**

 TZCBCFLG is shown in [Figure 18-143](#) and described in [Table 18-73](#).

 Return to the [Summary Table](#).

Trip Zone CBC Flag Register

**Figure 18-143. TZCBCFLG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
DCBEVT2	DCAEVT2	CBC6	CBC5	CBC4	CBC3	CBC2	CBC1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 18-73. TZCBCFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7	DCBEVT2	R	0h	Latched Status Flag for Digital Compare B Output Event 2 Trip Latch 0: Reading a 0 indicates that no trip has occurred on DCBEVT2. 1: Reading a 1 indicates a trip has occurred on the DCBEVT2 selected event. Reset type: SYSRSn
6	DCAEVT2	R	0h	Latched Status Flag for Digital Compare A Output Event 2 Trip Latch 0: Reading a 0 indicates that no trip has occurred on DCAEVT2. 1: Reading a 1 indicates a trip has occurred on the DCAEVT2 selected event. Reset type: SYSRSn
5	CBC6	R	0h	Latched Status Flag for CBC6 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC6. 1: Reading a 1 indicates a trip has occurred on the CBC6 selected event. Reset type: SYSRSn
4	CBC5	R	0h	Latched Status Flag for CBC5 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC5. 1: Reading a 1 indicates a trip has occurred on the CBC5 selected event. Reset type: SYSRSn
3	CBC4	R	0h	Latched Status Flag for CBC4 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC4. 1: Reading a 1 indicates a trip has occurred on the CBC4 selected event. Reset type: SYSRSn
2	CBC3	R	0h	Latched Status Flag for CBC3 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC3. 1: Reading a 1 indicates a trip has occurred on the CBC3 selected event. Reset type: SYSRSn

**Table 18-73. TZCBCFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	CBC2	R	0h	Latched Status Flag for CBC2 Trip Latch  0: Reading a 0 indicates that no trip has occurred on CBC2. 1: Reading a 1 indicates a trip has occurred on the CBC2 selected event. Reset type: SYSRSn
0	CBC1	R	0h	Latched Status Flag for CBC1 Trip Latch  0: Reading a 0 indicates that no trip has occurred on CBC1. 1: Reading a 1 indicates a trip has occurred on the CBC1 selected event. Reset type: SYSRSn

**18.16.2.51 TZOSTFLG Register (Offset = 95h) [reset = 0h]**

 TZOSTFLG is shown in [Figure 18-144](#) and described in [Table 18-74](#).

 Return to the [Summary Table](#).

Trip Zone OST Flag Register

**Figure 18-144. TZOSTFLG Register**

15		14		13		12		11		10		9		8	
RESERVED															
R-0-0h															
7		6		5		4		3		2		1		0	
DCBEVT1		DCAEVT1		OST6		OST5		OST4		OST3		OST2		OST1	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h	

**Table 18-74. TZOSTFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7	DCBEVT1	R	0h	Latched Status Flag for Digital Compare B Output Event 1 Trip Latch  0: Reading a 0 indicates that no trip has occurred on DCBEVT1. 1: Reading a 1 indicates a trip has occurred on the DCBEVT1 selected event.  Reset type: SYSRSn
6	DCAEVT1	R	0h	Latched Status Flag for Digital Compare A Output Event 1 Trip Latch  0: Reading a 0 indicates that no trip has occurred on DCAEVT1. 1: Reading a 1 indicates a trip has occurred on the DCAEVT1 selected event.  Reset type: SYSRSn
5	OST6	R	0h	Latched Status Flag for OST6 Trip Latch  0: Reading a 0 indicates that no trip has occurred on OST6. 1: Reading a 1 indicates a trip has occurred on the OST6 selected event.  Reset type: SYSRSn
4	OST5	R	0h	Latched Status Flag for OST5 Trip Latch  0: Reading a 0 indicates that no trip has occurred on OST5. 1: Reading a 1 indicates a trip has occurred on the OST5 selected event.  Reset type: SYSRSn
3	OST4	R	0h	Latched Status Flag for OST4 Trip Latch  0: Reading a 0 indicates that no trip has occurred on OST4. 1: Reading a 1 indicates a trip has occurred on the OST4 selected event.  Reset type: SYSRSn
2	OST3	R	0h	Latched Status Flag for OST3 Trip Latch  0: Reading a 0 indicates that no trip has occurred on OST3. 1: Reading a 1 indicates a trip has occurred on the OST3 selected event.  Reset type: SYSRSn

**Table 18-74. TZOSTFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	OST2	R	0h	Latched Status Flag for OST2 Trip Latch  0: Reading a 0 indicates that no trip has occurred on OST2. 1: Reading a 1 indicates a trip has occurred on the OST2 selected event. Reset type: SYSRSn
0	OST1	R	0h	Latched Status Flag for OST1 Trip Latch  0: Reading a 0 indicates that no trip has occurred on OST1. 1: Reading a 1 indicates a trip has occurred on the OST1 selected event. Reset type: SYSRSn

**18.16.2.52 TZCLR Register (Offset = 97h) [reset = 0h]**

 TZCLR is shown in [Figure 18-145](#) and described in [Table 18-75](#).

 Return to the [Summary Table](#).

Trip Zone Clear Register

**Figure 18-145. TZCLR Register**

15		14		13		12		11		10		9		8	
CBCPULSE				RESERVED											
R/W-0h				R-0-0h											
7		6		5		4		3		2		1		0	
RESERVED		DCBEVT2		DCBEVT1		DCAEVT2		DCAEVT1		OST		CBC		INT	
R-0-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h	

**Table 18-75. TZCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	CBCPULSE	R/W	0h	Clear Pulse for Cycle-By-Cycle (CBC) Trip Latch  This bit field determines which pulse clears the CBC trip latch.  00: CTR = zero pulse clears CBC trip latch. (Same as legacy designs.) 01: CTR = PRD pulse clears CBC trip latch. 10: CTR = zero or CTR = PRD pulse clears CBC trip latch. 11: CBC trip latch is not cleared Reset type: SYSRSn
13-7	RESERVED	R-0	0h	Reserved
6	DCBEVT2	R-0/W1S	0h	Clear Flag for Digital Compare Output B Event 2  0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 clears the DCBEVT2 event trip condition. Reset type: SYSRSn
5	DCBEVT1	R-0/W1S	0h	Clear Flag for Digital Compare Output B Event 1  0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 clears the DCBEVT1 event trip condition. Reset type: SYSRSn
4	DCAEVT2	R-0/W1S	0h	Clear Flag for Digital Compare Output A Event 2  0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 clears the DCAEVT2 event trip condition. Reset type: SYSRSn
3	DCAEVT1	R-0/W1S	0h	Clear Flag for Digital Compare Output A Event 1  0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 clears the DCAEVT1 event trip condition. Reset type: SYSRSn
2	OST	R-0/W1S	0h	Clear Flag for One-Shot Trip (OST) Latch  0: Has no effect. Always reads back a 0. 1: Clears this Trip (set) condition. Reset type: SYSRSn
1	CBC	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC) Trip Latch  0: Has no effect. Always reads back a 0. 1: Clears this Trip (set) condition. Reset type: SYSRSn



**Table 18-75. TZCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	INT	R-0/W1S	0h	Global Interrupt Clear Flag  0: Has no effect. Always reads back a 0. 1: Clears the trip-interrupt flag for this ePWM module (TZFLG[INT]).  NOTE: No further EPWMx_TZINT PIE interrupts will be generated until the flag is cleared. If the TZFLG[INT] bit is cleared and any of the other flag bits are set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts.  Reset type: SYSRSn

**18.16.2.53 TZCBCCLR Register (Offset = 98h) [reset = 0h]**

 TZCBCCLR is shown in [Figure 18-146](#) and described in [Table 18-76](#).

 Return to the [Summary Table](#).

Trip Zone CBC Clear Register

**Figure 18-146. TZCBCCLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
DCBEVT2	DCAEVT2	CBC6	CBC5	CBC4	CBC3	CBC2	CBC1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 18-76. TZCBCCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7	DCBEVT2	R-0/W1S	0h	Clear Flag for Digital Compare Output B Event 2 selected for CBC 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[DCBEVT2] bit. Reset type: SYSRSn
6	DCAEVT2	R-0/W1S	0h	Clear Flag for Digital Compare Output A Event 2 selected for CBC 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[DCAEVT2] bit. Reset type: SYSRSn
5	CBC6	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC6) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC6] bit. Reset type: SYSRSn
4	CBC5	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC5) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC5] bit. Reset type: SYSRSn
3	CBC4	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC4) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC4] bit. Reset type: SYSRSn
2	CBC3	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC3) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC3] bit. Reset type: SYSRSn
1	CBC2	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC2) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC2] bit. Reset type: SYSRSn

**Table 18-76. TZCBCCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	CBC1	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC1) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC1] bit. Reset type: SYSRSn

**18.16.2.54 TZOSTCLR Register (Offset = 99h) [reset = 0h]**

TZOSTCLR is shown in [Figure 18-147](#) and described in [Table 18-77](#).

Return to the [Summary Table](#).

Trip Zone OST Clear Register

**Figure 18-147. TZOSTCLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
DCBEVT1	DCAEVT1	OST6	OST5	OST4	OST3	OST2	OST1
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 18-77. TZOSTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7	DCBEVT1	R-0/W1S	0h	Clear Flag for Digital Compare Output B Event 1 selected for OST 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[DCBEVT1] bit. Reset type: SYSRSn
6	DCAEVT1	R-0/W1S	0h	Clear Flag for Digital Compare Output A Event 1 selected for OST 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[DCAEVT1] bit. Reset type: SYSRSn
5	OST6	R-0/W1S	0h	Clear Flag for Oneshot (OST6) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST6] bit. Reset type: SYSRSn
4	OST5	R-0/W1S	0h	Clear Flag for Oneshot (OST5) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST5] bit. Reset type: SYSRSn
3	OST4	R-0/W1S	0h	Clear Flag for Oneshot (OST4) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST4] bit. Reset type: SYSRSn
2	OST3	R-0/W1S	0h	Clear Flag for Oneshot (OST3) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST3] bit. Reset type: SYSRSn
1	OST2	R-0/W1S	0h	Clear Flag for Oneshot (OST2) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST2] bit. Reset type: SYSRSn

**Table 18-77. TZOSTCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	OST1	R-0/W1S	0h	Clear Flag for Oneshot (OST1) Trip Latch  0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST1] bit. Reset type: SYSRSn

**18.16.2.55 TZFRC Register (Offset = 9Bh) [reset = 0h]**

 TZFRC is shown in [Figure 18-148](#) and described in [Table 18-78](#).

 Return to the [Summary Table](#).

Trip Zone Force Register

**Figure 18-148. TZFRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	RESERVED
R-0-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0-0h

**Table 18-78. TZFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6	DCBEVT2	R-0/W1S	0h	Force Flag for Digital Compare Output B Event 2  0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 forces the DCBEVT2 event trip condition and sets the TZFLG[DCBEVT2] bit. Reset type: SYSRSn
5	DCBEVT1	R-0/W1S	0h	Force Flag for Digital Compare Output B Event 1  0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 forces the DCBEVT1 event trip condition and sets the TZFLG[DCBEVT1] bit. Reset type: SYSRSn
4	DCAEVT2	R-0/W1S	0h	Force Flag for Digital Compare Output A Event 2  0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 forces the DCAEVT2 event trip condition and sets the TZFLG[DCAEVT2] bit. Reset type: SYSRSn
3	DCAEVT1	R-0/W1S	0h	Force Flag for Digital Compare Output A Event 1  0: Writing 0 has no effect. This bit always reads back 0 1: Writing 1 forces the DCAEVT1 event trip condition and sets the TZFLG[DCAEVT1] bit. Reset type: SYSRSn
2	OST	R-0/W1S	0h	Force a One-Shot Trip Event via Software  0: Writing of 0 is ignored. Always reads back a 0. 1: Forces a one-shot trip event and sets the TZFLG[OST] bit. Reset type: SYSRSn
1	CBC	R-0/W1S	0h	Force a Cycle-by-Cycle Trip Event via Software  0: Writing of 0 is ignored. Always reads back a 0. 1: Forces a cycle-by-cycle trip event and sets the TZFLG[CBC] bit. Reset type: SYSRSn
0	RESERVED	R-0	0h	Reserved

### 18.16.2.56 ETSEL Register (Offset = A4h) [reset = 0h]

ETSEL is shown in [Figure 18-149](#) and described in [Table 18-79](#).

Return to the [Summary Table](#).

Event Trigger Selection Register

**Figure 18-149. ETSEL Register**

15		14		13		12		11		10		9		8	
SOCBEN		SOCBSEL						SOCAEN		SOCASEL					
R/W-0h		R/W-0h						R/W-0h		R/W-0h					
7		6		5		4		3		2		1		0	
RESERVED		INTSELCMP		SOCBSELCMP		SOCASELCMP		INTEN		INTSEL					
R-0-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h					

**Table 18-79. ETSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOCBEN	R/W	0h	Enable the ADC Start of Conversion B (EPWMxSOCB) Pulse 0: Disable EPWMxSOCB. 1: Enable EPWMxSOCB pulse. Reset type: SYSRSn
14-12	SOCBSEL	R/W	0h	EPWMxSOCB Selection Options These bits determine when a EPWMxSOCB pulse will be generated. 000: Enable DCBEVT1.soc event 001: Enable event time-base counter equal to zero. (TBCTR = 0x00) 010: Enable event time-base counter equal to period (TBCTR = TBPRD) 011: Enable event time-base counter equal to zero or period (TBCTR = 0x00 or TBCTR = TBPRD). This mode is useful in up-down count mode. 100: Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing 101: Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing 110: Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing 111: Enable event: time-base counter equal to CMPB when the timer is decrementing or CMPD when the timer is decrementing (*) Event selected is determined by SOCBSELCMP bit. Reset type: SYSRSn
11	SOCAEN	R/W	0h	Enable the ADC Start of Conversion A (EPWMxSOCA) Pulse 0: Disable EPWMxSOCA. 1: Enable EPWMxSOCA pulse. Reset type: SYSRSn

**Table 18-79. ETSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-8	SOCASEL	R/W	0h	EPWMxSOCA Selection Options  These bits determine when a EPWMxSOCA pulse will be generated.  000: Enable DCAEVT1.soc event 001: Enable event time-base counter equal to zero. (TBCTR = 0x00) 010: Enable event time-base counter equal to period (TBCTR = TBPRD) 011: Enable event time-base counter equal to zero or period (TBCTR = 0x00 or TBCTR = TBPRD). This mode is useful in up-down count mode. 100: Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing 101: Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing 110: Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing 111: Enable event: time-base counter equal to CMPB when the timer is decrementing or CMPD when the timer is decrementing (*) Event selected is determined by SOCASELCMP bit.  Reset type: SYSRSn
7	RESERVED	R-0	0h	Reserved
6	INTSELCMP	R/W	0h	EPWMxINT Compare Register Selection Options  0: Enable event time-base counter equal to CMPA when the timer is incrementing / Enable event time-base counter equal to CMPA when the timer is decrementing / Enable event: time-base counter equal to CMPB when the timer is incrementing / Enable event: time-base counter equal to CMPB when the timer is decrementing to INTSEL selection mux.  1: Enable event time-base counter equal to CMPC when the timer is incrementing / Enable event time-base counter equal to CMPC when the timer is decrementing / Enable event: time-base counter equal to CMPD when the timer is incrementing / Enable event: time-base counter equal to CMPD when the timer is decrementing to INTSEL selection mux.  Reset type: SYSRSn
5	SOCBSELCMP	R/W	0h	EPWMxSOCB Compare Register Selection Options  0: Enable event time-base counter equal to CMPA when the timer is incrementing / Enable event time-base counter equal to CMPA when the timer is decrementing / Enable event: time-base counter equal to CMPB when the timer is incrementing / Enable event: time-base counter equal to CMPB when the timer is decrementing to SOCBSEL selection mux.  1: Enable event time-base counter equal to CMPC when the timer is incrementing / Enable event time-base counter equal to CMPC when the timer is decrementing / Enable event: time-base counter equal to CMPD when the timer is incrementing / Enable event: time-base counter equal to CMPD when the timer is decrementing to SOCBSEL selection mux.  Reset type: SYSRSn



**Table 18-79. ETSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	SOCASELCMP	R/W	0h	<p>EPWMxSOCA Compare Register Selection Options</p> <p>0: Enable event time-base counter equal to CMPA when the timer is incrementing / Enable event time-base counter equal to CMPA when the timer is decrementing / Enable event: time-base counter equal to CMPB when the timer is incrementing / Enable event: time-base counter equal to CMPB when the timer is decrementing to SOCASEL selection mux.</p> <p>1: Enable event time-base counter equal to CMPC when the timer is incrementing / Enable event time-base counter equal to CMPC when the timer is decrementing / Enable event: time-base counter equal to CMPD when the timer is incrementing / Enable event: time-base counter equal to CMPD when the timer is decrementing to SOCASEL selection mux.</p> <p>Reset type: SYSRSn</p>
3	INTEN	R/W	0h	<p>Enable ePWM Interrupt (EPWMx_INT) Generation</p> <p>0: Disable EPWMx_INT generation</p> <p>1: Enable EPWMx_INT generation</p> <p>Reset type: SYSRSn</p>
2-0	INTSEL	R/W	0h	<p>ePWM Interrupt (EPWMx_INT) Selection Options</p> <p>000: Reserved</p> <p>001: Enable event time-base counter equal to zero. (TBCTR = 0x00)</p> <p>010: Enable event time-base counter equal to period (TBCTR = TBPRD)</p> <p>011: Enable event time-base counter equal to zero or period (TBCTR = 0x00 or TBCTR = TBPRD). This mode is useful in up-down count mode.</p> <p>100: Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing</p> <p>101: Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing</p> <p>110: Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing</p> <p>111: Enable event: time-base counter equal to CMPB when the timer is decrementing or CMPD when the timer is decrementing (*) Event selected is determined by INTSELCMP bit.</p> <p>Reset type: SYSRSn</p>

**18.16.2.57 ETPS Register (Offset = A6h) [reset = 0h]**

 ETPS is shown in [Figure 18-150](#) and described in [Table 18-80](#).

 Return to the [Summary Table](#).

Event Trigger Pre-Scale Register

**Figure 18-150. ETPS Register**

15	14	13	12	11	10	9	8
SOCBCNT		SOCBPRD		SOCACNT		SOCAPRD	
R-0h		R/W-0h		R-0h		R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		SOCPSSEL	INTPSSEL	INTCNT		INTPRD	
R-0-0h		R/W-0h	R/W-0h	R-0h		R/W-0h	

**Table 18-80. ETPS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	SOCBCNT	R	0h	ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Counter Register  These bits indicate how many selected ETSEL[SOCBSEL] events have occurred:  00: No events have occurred. 01: 1 event has occurred. 10: 2 events have occurred. 11: 3 events have occurred. Reset type: SYSRSn
13-12	SOCBPRD	R/W	0h	ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Period Select  These bits determine how many selected ETSEL[SOCBSEL] events need to occur before an EPWMxSOCB pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCBEN] = 1). The SOCB pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCB] = 1). Once the SOCB pulse is generated, the ETPS[SOCBCNT] bits will automatically be cleared.  00: Disable the SOCB event counter. No EPWMxSOCB pulse will be generated 01: Generate the EPWMxSOCB pulse on the first event: ETPS[SOCBCNT] = 0,1 10: Generate the EPWMxSOCB pulse on the second event: ETPS[SOCBCNT] = 1,0 11: Generate the EPWMxSOCB pulse on the third event: ETPS[SOCBCNT] = 1,1 Reset type: SYSRSn
11-10	SOCACNT	R	0h	ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Counter Register  These bits indicate how many selected ETSEL[SOCASEL] events have occurred:  00: No events have occurred. 01: 1 event has occurred. 10: 2 events have occurred. 11: 3 events have occurred. Reset type: SYSRSn

**Table 18-80. ETPS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	SOCAPRD	R/W	0h	<p>ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Period Select</p> <p>These bits determine how many selected ETSEL[SOCASEL] events need to occur before an EPWMxSOCA pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCAEN] = 1). The SOCA pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCA] = 1). Once the SOCA pulse is generated, the ETPS[SOCACNT] bits will automatically be cleared.</p> <p>00: Disable the SOCA event counter. No EPWMxSOCA pulse will be generated</p> <p>01: Generate the EPWMxSOCA pulse on the first event: ETPS[SOCACNT] = 0,1</p> <p>10: Generate the EPWMxSOCA pulse on the second event: ETPS[SOCACNT] = 1,0</p> <p>11: Generate the EPWMxSOCA pulse on the third event: ETPS[SOCACNT] = 1,1</p> <p>Reset type: SYSRSn</p>
7-6	RESERVED	R-0	0h	Reserved
5	SOCPSSEL	R/W	0h	<p>EPWMxSOC A/B Pre-Scale Selection Bits</p> <p>0: Selects ETPS [SOCACNT/SOCBCNT] and [SOCAPRD/SOCBPRD] registers to determine frequency of events (interrupt once every 0-3 events).</p> <p>1: Selects ETSOCPS [SOCACNT2/SOCBCNT2] and [SOCAPRD2/SOCBPRD2] registers to determine frequency of events (interrupt once every 0-15 events).</p> <p>Reset type: SYSRSn</p>
4	INTPSSEL	R/W	0h	<p>EPWMxINTn Pre-Scale Selection Bits</p> <p>0: Selects ETPS [INTCNT, and INTPRD] registers to determine frequency of events (interrupt once every 0-3 events).</p> <p>1: Selects ETINTPS [ INTCNT2, and INTPRD2 ] registers to determine frequency of events (interrupt once every 0-15 events).</p> <p>Reset type: SYSRSn</p>
3-2	INTCNT	R	0h	<p>ePWM Interrupt Event (EPWMx_INT) Counter Register</p> <p>These bits indicate how many selected ETSEL[INTSEL] events have occurred. These bits are automatically cleared when an interrupt pulse is generated. If interrupts are disabled, ETSEL[INT] = 0 or the interrupt flag is set, ETFLG[INT] = 1, the counter will stop counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].</p> <p>00: No events have occurred.</p> <p>01: 1 event has occurred.</p> <p>10: 2 events have occurred.</p> <p>11: 3 events have occurred.</p> <p>Reset type: SYSRSn</p>

**Table 18-80. ETPS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	INTPRD	R/W	0h	<p>ePWM Interrupt (EPWMx_INT) Period Select</p> <p>These bits determine how many selected ETSEL[INTSEL] events need to occur before an interrupt is generated. To be generated, the interrupt must be enabled (ETSEL[INT] = 1). If the interrupt status flag is set from a previous interrupt (ETFLG[INT] = 1) then no interrupt will be generated until the flag is cleared via the ETCLR[INT] bit. This allows for one interrupt to be pending while another is still being serviced. Once the interrupt is generated, the ETPS[INTCNT] bits will automatically be cleared.</p> <p>Writing a INTPRD value that is the same as the current counter value will trigger an interrupt if it is enabled and the status flag is clear.</p> <p>Writing a INTPRD value that is less than the current counter value will result in an undefined state. If a counter event occurs at the same instant as a new zero or non-zero INTPRD value is written, the counter is incremented.</p> <p>00: Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored.</p> <p>01: Generate an interrupt on the first event INTCNT = 01 (first event)</p> <p>10: Generate interrupt on ETPS[INTCNT] = 1,0 (second event)</p> <p>11: Generate interrupt on ETPS[INTCNT] = 1,1 (third event)</p> <p>Reset type: SYSRSn</p>

### 18.16.2.58 ETFLG Register (Offset = A8h) [reset = 0h]

ETFLG is shown in [Figure 18-151](#) and described in [Table 18-81](#).

Return to the [Summary Table](#).

Event Trigger Flag Register

**Figure 18-151. ETFLG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				SOCB	SOCA	RESERVED	INT
R-0-0h				R-0h	R-0h	R-0-0h	R-0h

**Table 18-81. ETFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3	SOCB	R	0h	Latched ePWM ADC Start-of-Conversion A (EPWMxSOCB) Status Flag  Unlike the ETFLG[INT] flag, the EPWMxSOCB output will continue to pulse even if the flag bit is set.  0: Indicates no event occurred 1: Indicates that a start of conversion pulse was generated on EPWMxSOCB. The EPWMxSOCB output will continue to be generated even if the flag bit is set. Reset type: SYSRSn
2	SOCA	R	0h	Latched ePWM ADC Start-of-Conversion A (EPWMxSOCA) Status Flag  Unlike the ETFLG[INT] flag, the EPWMxSOCA output will continue to pulse even if the flag bit is set.  0: Indicates no event occurred 1: Indicates that a start of conversion pulse was generated on EPWMxSOCA. The EPWMxSOCA output will continue to be generated even if the flag bit is set. Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	INT	R	0h	Latched ePWM Interrupt (EPWMx_INT) Status Flag  0: Indicates no event occurred 1: Indicates that an ePWMx interrupt (EPWMx_INT) was generated. No further interrupts will be generated until the flag bit is cleared. Up to one interrupt can be pending while the ETFLG[INT] bit is still set. If an interrupt is pending, it will not be generated until after the ETFLG[INT] bit is cleared. Reset type: SYSRSn

**18.16.2.59 ETCLR Register (Offset = AAh) [reset = 0h]**

 ETCLR is shown in [Figure 18-152](#) and described in [Table 18-82](#).

 Return to the [Summary Table](#).

Event Trigger Clear Register

**Figure 18-152. ETCLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				SOCB	SOCA	RESERVED	INT
R-0-0h				R-0/W1S-0h	R-0/W1S-0h	R-0-0h	R-0/W1S-0h

**Table 18-82. ETCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3	SOCB	R-0/W1S	0h	ePWM ADC Start-of-Conversion A (EPWMxSOCB) Flag Clear Bit 0: Writing a 0 has no effect. Always reads back a 0 1: Clears the ETFLG[SOCB] flag bit Reset type: SYSRSn
2	SOCA	R-0/W1S	0h	ePWM ADC Start-of-Conversion A (EPWMxSOCA) Flag Clear Bit 0: Writing a 0 has no effect. Always reads back a 0 1: Clears the ETFLG[SOCA] flag bit Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	INT	R-0/W1S	0h	ePWM Interrupt (EPWMx_INT) Flag Clear Bit 0: Writing a 0 has no effect. Always reads back a 0 1: Clears the ETFLG[INT] flag bit and enable further interrupts pulses to be generated Reset type: SYSRSn

**18.16.2.60 ETFRC Register (Offset = ACh) [reset = 0h]**

ETFRC is shown in [Figure 18-153](#) and described in [Table 18-83](#).

Return to the [Summary Table](#).

Event Trigger Force Register

**Figure 18-153. ETFRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				SOCB	SOCA	RESERVED	INT
R-0-0h				R-0/W1S-0h	R-0/W1S-0h	R-0-0h	R-0/W1S-0h

**Table 18-83. ETFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3	SOCB	R-0/W1S	0h	<p>SOCB Force Bit</p> <p>The SOCB pulse will only be generated if the event is enabled in the ETSEL register. The ETFLG[SOCB] flag bit will be set regardless.</p> <p>0: Writing 0 to this bit will be ignored. Always reads back a 0.</p> <p>1: Generates a pulse on EPWMxSOCB and set the SOCBFLG bit. This bit is used for test purposes.</p> <p>Reset type: SYSRSn</p>
2	SOCA	R-0/W1S	0h	<p>SOCA Force Bit</p> <p>The SOCA pulse will only be generated if the event is enabled in the ETSEL register. The ETFLG[SOCA] flag bit will be set regardless.</p> <p>0: Writing 0 to this bit will be ignored. Always reads back a 0.</p> <p>1: Generates a pulse on EPWMxSOCA and set the SOCAFLG bit. This bit is used for test purposes.</p> <p>Reset type: SYSRSn</p>
1	RESERVED	R-0	0h	Reserved
0	INT	R-0/W1S	0h	<p>INT Force Bit</p> <p>The interrupt will only be generated if the event is enabled in the ETSEL register. The INT flag bit will be set regardless.</p> <p>0: Writing 0 to this bit will be ignored. Always reads back a 0.</p> <p>1: Generates an interrupt on EPWMxINT and set the INT flag bit. This bit is used for test purposes.</p> <p>Reset type: SYSRSn</p>

**18.16.2.61 ETINTPS Register (Offset = AEh) [reset = 0h]**

 ETINTPS is shown in [Figure 18-154](#) and described in [Table 18-84](#).

 Return to the [Summary Table](#).

Event-Trigger Interrupt Pre-Scale Register

**Figure 18-154. ETINTPS Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
INTCNT2				INTPRD2			
R-0h				R/W-0h			

**Table 18-84. ETINTPS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-4	INTCNT2	R	0h	EPWMxINT Counter 2  When ETPS[INTPSEL]=1, these bits indicate how many selected events have occurred:  0000: No events 0001: 1 event 0010: 2 events 0011: 3 events 0100: 4 events ... 1111: 15 events Reset type: SYSRSn
3-0	INTPRD2	R/W	0h	EPWMxINT Period 2 Select  When ETPS[INTPSEL] = 1, these bits select how many selected events need to occur before an interrupt is generated:  0000: Disable counter 0001: Generate interrupt on INTCNT = 1 (first event) 0010: Generate interrupt on INTCNT = 2 (second event) 0011: Generate interrupt on INTCNT = 3 (third event) 0100: Generate interrupt on INTCNT = 4 (fourth event) ... 1111: Generate interrupt on INTCNT = 15 (fifteenth event) Reset type: SYSRSn



### 18.16.2.62 ETSOCPS Register (Offset = B0h) [reset = 0h]

ETSOCPS is shown in [Figure 18-155](#) and described in [Table 18-85](#).

Return to the [Summary Table](#).

Event-Trigger SOC Pre-Scale Register

**Figure 18-155. ETSOCPS Register**

15	14	13	12	11	10	9	8
SOCBCNT2				SOCBPRD2			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
SOCACNT2				SOCAPRD2			
R-0h				R/W-0h			

**Table 18-85. ETSOCPS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	SOCBCNT2	R	0h	<p>EPWMxSOCB Counter 2</p> <p>When ETPS[SOCPSSEL] = 1, these bits indicate how many selected events have occurred:</p> <p>0000: No events            0001: 1 event            0010: 2 events            0011: 3 events            0100: 4 events            ...            1111: 15 events</p> <p>Reset type: SYSRSn</p>
11-8	SOCBPRD2	R/W	0h	<p>EPWMxSOCB Period 2 Select</p> <p>When ETPS[SOCPSSEL] = 1, these bits select how many selected event need to occur before an SOCB pulse is generated:</p> <p>0000: Disable counter            0001: Generate interrupt on SOCBCNT2 = 1 (first event)            0010: Generate interrupt on SOCBCNT2 = 2 (second event)            0011: Generate interrupt on SOCBCNT2 = 3 (third event)            0100: Generate interrupt on SOCBCNT2 = 4 (fourth event)            ...            1111: Generate interrupt on SOCBCNT2 = 15 (fifteenth event)</p> <p>Reset type: SYSRSn</p>
7-4	SOCACNT2	R	0h	<p>EPWMxSOCA Counter 2</p> <p>When ETPS[SOCPSSEL] = 1, these bits indicate how many selected events have occurred:</p> <p>0000: No events            0001: 1 event            0010: 2 events            0011: 3 events            0100: 4 events            ...            1111: 15 events</p> <p>Reset type: SYSRSn</p>

**Table 18-85. ETSOCPS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	SOCAPRD2	R/W	0h	<p>EPWMxSOCA Period 2 Select</p> <p>When ETPS[SOCPSSEL] = 1, these bits select how many selected event need to occur before an SOCA pulse is generated:</p> <p>0000: Disable counter            0001: Generate interrupt on SOCACNT2 = 1 (first event)            0010: Generate interrupt on SOCACNT2 = 2 (second event)            0011: Generate interrupt on SOCACNT2 = 3 (third event)            0100: Generate interrupt on SOCACNT2 = 4 (fourth event)            ...            1111: Generate interrupt on SOCACNT2 = 15 (fifteenth event)</p> <p>Reset type: SYSRSn</p>

### 18.16.2.63 ETCNTINITCTL Register (Offset = B2h) [reset = 0h]

ETCNTINITCTL is shown in [Figure 18-156](#) and described in [Table 18-86](#).

Return to the [Summary Table](#).

Event-Trigger Counter Initialization Control Register

**Figure 18-156. ETCNTINITCTL Register**

15		14		13		12		11		10		9		8	
SOCBINITEN		SOCAINITEN		INTINITEN		SOCBINITFRC		SOCAINITFRC		INTINITFRC		RESERVED			
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R-0-0h			
7		6		5		4		3		2		1		0	
RESERVED															
R-0-0h															

**Table 18-86. ETCNTINITCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOCBINITEN	R/W	0h	EPWMxSOCB Counter 2 Initialization Enable  0: Has no effect. 1: Enable initialization of EPWMxSOCB counter with contents of ETCNTINIT[SOCBINIT] on a SYNC event or software force. Reset type: SYSRSn
14	SOCAINITEN	R/W	0h	EPWMxSOCA Counter 2 Initialization Enable  0: Has no effect. 1: Enable initialization of EPWMxSOCA counter with contents of ETCNTINIT[SOCAINIT] on a SYNC event or software force. Reset type: SYSRSn
13	INTINITEN	R/W	0h	EPWMxINT Counter 2 Initialization Enable  0: Has no effect. 1: Enable initialization of EPWMxINT counter 2 with contents of ETCNTINIT[INTINIT] on a SYNC event or software force. Reset type: SYSRSn
12	SOCBINITFRC	R/W	0h	EPWMxSOCB Counter 2 Initialization Force  0: Has no effect. 1: This bit forces the ET EPWMxSOCB counter to be initialized with the contents of ETCNTINIT[SOCBINIT]. Reset type: SYSRSn
11	SOCAINITFRC	R/W	0h	EPWMxSOCA Counter 2 Initialization Force  0: Has no effect. 1: This bit forces the ET EPWMxSOCA counter to be initialized with the contents of ETCNTINIT[SOCAINIT]. Reset type: SYSRSn
10	INTINITFRC	R/W	0h	EPWMxINT Counter 2 Initialization Force  0: Has no effect. 1: This bit forces the ET EPWMxINT counter to be initialized with the contents of ETCNTINIT[INTINIT]. Reset type: SYSRSn
9-0	RESERVED	R-0	0h	Reserved

**18.16.2.64 ETCNTINIT Register (Offset = B4h) [reset = 0h]**

ETCNTINIT is shown in [Figure 18-157](#) and described in [Table 18-87](#).

Return to the [Summary Table](#).

Event-Trigger Counter Initialization Register

**Figure 18-157. ETCNTINIT Register**

15	14	13	12	11	10	9	8
RESERVED				SOCBINIT			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
SOCAINIT				INTINIT			
R/W-0h				R/W-0h			

**Table 18-87. ETCNTINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-8	SOCBINIT	R/W	0h	EPWMxSOCB Counter 2 Initialization Bits The ET EPWMxSOCB counter is initialized with the contents of this register on an ePWM SYNC event or a software force. Reset type: SYSRSn
7-4	SOCAINIT	R/W	0h	EPWMxSOCA Counter 2 Initialization Bits The ET EPWMxSOCA counter is initialized with the contents of this register on an ePWM SYNC event or a software force. Reset type: SYSRSn
3-0	INTINIT	R/W	0h	EPWMxINT Counter 2 Initialization Bits The ET EPWMxINT counter is initialized with the contents of this register on an ePWM SYNC event or a software force. Reset type: SYSRSn

### 18.16.2.65 DCTRIPSEL Register (Offset = C0h) [reset = 0h]

DCTRIPSEL is shown in [Figure 18-158](#) and described in [Table 18-88](#).

Return to the [Summary Table](#).

Digital Compare Trip Select Register

**Figure 18-158. DCTRIPSEL Register**

15	14	13	12	11	10	9	8
DCBLCOMPSEL				DCBHCOMPSEL			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
DCALCOMPSEL				DCAHCOMPSEL			
R/W-0h				R/W-0h			

**Table 18-88. DCTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	DCBLCOMPSEL	R/W	0h	Digital Compare B Low Input Select Bits  0000: TRIPIN1 and (TZ1 input) 0001: TRIPIN2 and (TZ2 input) 0010: TRIPIN3 and (TZ3 input) 0011: TRIPIN4 ... 1011: TRIPIN12 1100: Reserved 1101: TRIPIN14 1110: TRIPIN15 1111: Trip combination input (all trip inputs selected by DCBLTRIPSEL register ORed together) Reset type: SYSRSn
11-8	DCBHCOMPSEL	R/W	0h	Digital Compare B High Input Select Bits  0000: TRIPIN1 and (TZ1 input) 0001: TRIPIN2 and (TZ2 input) 0010: TRIPIN3 and (TZ3 input) 0011: TRIPIN4 ... 1011: TRIPIN12 1100: Reserved 1101: TRIPIN14 1110: TRIPIN15 1111: Trip combination input (all trip inputs selected by DCBHTRIPSEL register ORed together) Reset type: SYSRSn

**Table 18-88. DCTRIPSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-4	DCALCOMPSEL	R/W	0h	Digital Compare A Low Input Select Bits 0000: TRIPIN1 and (TZ1 input) 0001: TRIPIN2 and (TZ2 input) 0010: TRIPIN3 and (TZ3 input) 0011: TRIPIN4 ... 1011: TRIPIN12 1100: Reserved 1101: TRIPIN14 1110: TRIPIN15 1111: Trip combination input (all trip inputs selected by DCALTRIPSEL register ORed together) Reset type: SYSRSn
3-0	DCAHCOMPSEL	R/W	0h	Digital Compare A High Input Select Bits 0000: TRIPIN1 and (TZ1 input) 0001: TRIPIN2 and (TZ2 input) 0010: TRIPIN3 and (TZ3 input) 0011: TRIPIN4 ... 1011: TRIPIN12 1100: Reserved 1101: TRIPIN14 1110: TRIPIN15 1111: Trip combination input (all trip inputs selected by DCAHTRIPSEL register ORed together) Reset type: SYSRSn

**18.16.2.66 DCACTL Register (Offset = C3h) [reset = 0h]**

DCACTL is shown in [Figure 18-159](#) and described in [Table 18-89](#).

Return to the [Summary Table](#).

Digital Compare A Control Register

**Figure 18-159. DCACTL Register**

15	14	13	12	11	10	9	8
				RESERVED		EVT2FRCSYN CSEL	EVT2SRCSEL
				R-0-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
				EVT1SYNCE	EVT1SOCE	EVT1FRCSYN CSEL	EVT1SRCSEL
				R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 18-89. DCACTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11-10	RESERVED	R-0	0h	Reserved
9	EVT2FRCSYNSEL	R/W	0h	DCAEVT2 Force Synchronization Signal Select 0: Source is passed through asynchronously 1: Source is synchronized with EPWMCLK Reset type: SYSRSn
8	EVT2SRCSEL	R/W	0h	DCAEVT2 Source Signal Select 0: Source Is DCAEVT2 Signal 1: Source Is DCEVTFILT Signal Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6-5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	EVT1SYNCE	R/W	0h	DCAEVT1 SYNC, Enable/Disable 0: SYNC Generation Disabled 1: SYNC Generation Enabled Reset type: SYSRSn
2	EVT1SOCE	R/W	0h	DCAEVT1 SOC, Enable/Disable 0: SOC Generation Disabled 1: SOC Generation Enabled Reset type: SYSRSn
1	EVT1FRCSYNSEL	R/W	0h	DCAEVT1 Force Synchronization Signal Select 0: Source is synchronized with EPWMCLK 1: Source is passed through asynchronously Reset type: SYSRSn

**Table 18-89. DCACTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	EVT1SRCSEL	R/W	0h	DCAEVT1 Source Signal Select 0: Source Is DCAEVT1 Signal 1: Source Is DCEVTFILT Signal Reset type: SYSRSn



### 18.16.2.67 DCBCTL Register (Offset = C4h) [reset = 0h]

DCBCTL is shown in [Figure 18-160](#) and described in [Table 18-90](#).

Return to the [Summary Table](#).

Digital Compare B Control Register

**Figure 18-160. DCBCTL Register**

15	14	13	12	11	10	9	8
				RESERVED		EVT2FRCSYN CSEL	EVT2SRCSEL
				R-0-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
				EVT1SYNCE	EVT1SOCE	EVT1FRCSYN CSEL	EVT1SRCSEL
				R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 18-90. DCBCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11-10	RESERVED	R-0	0h	Reserved
9	EVT2FRCSYNSEL	R/W	0h	DCBEVT2 Force Synchronization Signal Select 0: Source is synchronized with EPWMCLK 1: Source is passed through asynchronously Reset type: SYSRSn
8	EVT2SRCSEL	R/W	0h	DCBEVT2 Source Signal Select 0: Source Is DCBEVT2 Signal 1: Source Is DCEVTFILT Signal Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6-5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	EVT1SYNCE	R/W	0h	DCBEVT1 SYNC, Enable/Disable 0: SYNC Generation Disabled 1: SYNC Generation Enabled Reset type: SYSRSn
2	EVT1SOCE	R/W	0h	DCBEVT1 SOC, Enable/Disable 0: SOC Generation Disabled 1: SOC Generation Enabled Reset type: SYSRSn
1	EVT1FRCSYNSEL	R/W	0h	DCBEVT1 Force Synchronization Signal Select 0: Source is synchronized with EPWMCLK 1: Source is passed through asynchronously Reset type: SYSRSn

**Table 18-90. DCBCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	EVT1SRCSEL	R/W	0h	DCBEVT1 Source Signal Select 0: Source Is DCBEVT1 Signal 1: Source Is DCEVTFILT Signal Reset type: SYSRSn

### 18.16.2.68 DCFCTL Register (Offset = C7h) [reset = 0h]

DCFCTL is shown in [Figure 18-161](#) and described in [Table 18-91](#).

Return to the [Summary Table](#).

Digital Compare Filter Control Register

**Figure 18-161. DCFCTL Register**

15	14	13	12	11	10	9	8
EDGESTATUS			EDGECOUNT			EDGEMODE	
R-0h			R/W-0h			R/W-0h	
7	6	5	4	3	2	1	0
RESERVED	EDGEFILTSEL	PULSESEL		BLANKINV	BLANKE	SRCSEL	
R-0-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h	

**Table 18-91. DCFCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	EDGESTATUS	R	0h	Edge Status: These bits reflect the total number of edges currently captured. When the value matches the EDGECOUNT, the status bits are set to zero, and a TBCLK wide pulse is generated which can then be output on the DCEVTFILT signal. The edge counter can be reset by writing 000 to the EDGECOUNT value: Reset type: SYSRSn
12-10	EDGECOUNT	R/W	0h	Edge Count: These bits select how many edges to count before generating a TBCLK wide pulse on the DCEVTFILT signal: 000: no edges, reset current EDGESTATUS bits to 0,0,0 001: 1 edge 010: 2 edges 011: 3 edges 100: 4 edges 101: 5 edges 110: 6 edges 111: 7 edges Reset type: SYSRSn
9-8	EDGEMODE	R/W	0h	Edge Mode Select: 00: Low To High Edge 01: High To Low Edge 10: Both Edges 11: Reserved Reset type: SYSRSn
7	RESERVED	R-0	0h	Reserved
6	EDGEFILTSEL	R/W	0h	Edge Filter Select: 0: Edge Filter Not Selected 1: Edge Filter Selected Reset type: SYSRSn
5-4	PULSESEL	R/W	0h	Pulse Select For Blanking & Capture Alignment 00: Time-base counter equal to period (TBCTR = TBPRD) 01: Time-base counter equal to zero (TBCTR = 0x00) 10: Time-base counter equal to zero (TBCTR = 0x00) or period (TBCTR = TBPRD) 11: Reserved Reset type: SYSRSn

**Table 18-91. DCFCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	BLANKINV	R/W	0h	Blanking Window Inversion 0: Blanking window not inverted 1: Blanking window inverted Reset type: SYSRSn
2	BLANKE	R/W	0h	Blanking Window Enable/Disable 0: Blanking window is disabled 1: Blanking window is enabled Reset type: SYSRSn
1-0	SRCSEL	R/W	0h	Filter Block Signal Source Select 00: Source Is DCAEVT1 Signal 01: Source Is DCAEVT2 Signal 10: Source Is DCBEVT1 Signal 11: Source Is DCBEVT2 Signal Reset type: SYSRSn

### 18.16.2.69 DCCAPCTL Register (Offset = C8h) [reset = 0h]

DCCAPCTL is shown in [Figure 18-162](#) and described in [Table 18-92](#).

Return to the [Summary Table](#).

Digital Compare Capture Control Register

**Figure 18-162. DCCAPCTL Register**

15		14		13		12		11		10		9		8	
CAPMODE		CAPCLR		CAPSTS		RESERVED									
R/W-0h		R-0/W1S-0h		R-0h		R-0-0h									
7		6		5		4		3		2		1		0	
RESERVED												SHDWMODE		CAPE	
R-0-0h												R/W-0h		R/W-0h	

**Table 18-92. DCCAPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	CAPMODE	R/W	0h	<p>Counter Capture Mode</p> <p>0: When a DCEVTFILT occurs and the counter capture is enabled, then the current TBCNT value is captured in the active register. When the respective trip event occurs, further trip (capture) events are ignored until the next PRD_eq or CNT_zero event (as selected by the PULSESEL bit in the DCFCTL register) re-triggers the capture mechanism.</p> <p>If active mode is enabled, via SHDWMODE bit in DCCAPCTL register, CPU reads of this register will return the active register value.</p> <p>If shadow mode is enabled, via SHDWMODE bit in DCCAPCTL register, the active register is copied to the shadow register on the PRD_eq or CNT_zero event (whichever is selected by PULSESEL bit in DCFCTL register). CPU reads of this register will return the shadow register value.</p> <p>1: When a DCEVTFILT occurs and the counter capture is enabled, then the current TBCNT value is captured in the active register. When the respective trip event occurs - it will set the CAPSTS flag and further trip (capture) events are ignored until this bit is cleared. CAPSTS can be cleared by writing to CAPCLR bit in DCCAPCTL register and it re-triggers the capture mechanism.</p> <p>If active mode is enabled, via SHDWMODE bit in DCCAPCTL register, CPU reads of this register will return the active register value.</p> <p>If shadow mode is enabled, via SHDWMODE bit in DCCAPCTL register, the active register is copied to the shadow register on the PRD_eq or CNT_zero event (whichever is selected by PULSESEL bit in DCFCTL register). CPU reads of this register will return the shadow register value.</p> <p>Reset type: SYSRSn</p>
14	CAPCLR	R-0/W1S	0h	<p>DC Capture Latched Status Clear Flag</p> <p>0: Writing a 0 has no effect.</p> <p>1: Writing a 1 will clear this CAPSTS (set) condition.</p> <p>Reset type: SYSRSn</p>
13	CAPSTS	R	0h	<p>Latched Status Flag for Capture Event</p> <p>0: No DC capture event occurred.</p> <p>1: A DC capture event has occurred.</p> <p>Reset type: SYSRSn</p>
12-2	RESERVED	R-0	0h	Reserved

**Table 18-92. DCCAPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	SHDWMODE	R/W	0h	<p>TBCTR Counter Capture Shadow Select Mode</p> <p>0: Enable shadow mode. The DCCAP active register is copied to shadow register on a TBCTR = TBPRD or TBCTR = zero event as defined by the DCFCTL[PULSESEL] bit. CPU reads of the DCCAP register will return the shadow register contents.</p> <p>1: Active Mode. In this mode the shadow register is disabled. CPU reads from the DCCAP register will always return the active register contents.</p> <p>Reset type: SYSRSn</p>
0	CAPE	R/W	0h	<p>TBCTR Counter Capture Enable/Disable</p> <p>0: Disable the time-base counter capture.</p> <p>1: Enable the time-base counter capture.</p> <p>Reset type: SYSRSn</p>

**18.16.2.70 DCFOFFSET Register (Offset = C9h) [reset = 0h]**

DCFOFFSET is shown in [Figure 18-163](#) and described in [Table 18-93](#).

Return to the [Summary Table](#).

Digital Compare Filter Offset Register

**Figure 18-163. DCFOFFSET Register**

15	14	13	12	11	10	9	8
DCFOFFSET							
R/W-0h							
7	6	5	4	3	2	1	0
DCFOFFSET							
R/W-0h							

**Table 18-93. DCFOFFSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCFOFFSET	R/W	0h	<p>Blanking Window Offset</p> <p>These 16-bits specify the number of TBCLK cycles from the blanking window reference to the point when the blanking window is applied. The blanking window reference is either period or zero as defined by the DCFCTL[PULSESEL] bit. This offset register is shadowed and the active register is loaded at the reference point defined by DCFCTL[PULSESEL]. The offset counter is also initialized and begins to count down when the active register is loaded. When the counter expires, the blanking window is applied. If the blanking window is currently active, then the blanking window counter is restarted.</p> <p>Reset type: SYSRSn</p>

**18.16.2.71 DCFOFFSETCNT Register (Offset = CAh) [reset = 0h]**

DCFOFFSETCNT is shown in [Figure 18-164](#) and described in [Table 18-94](#).

Return to the [Summary Table](#).

Digital Compare Filter Offset Counter Register

**Figure 18-164. DCFOFFSETCNT Register**

15	14	13	12	11	10	9	8
DCFOFFSETCNT							
R-0h							
7	6	5	4	3	2	1	0
DCFOFFSETCNT							
R-0h							

**Table 18-94. DCFOFFSETCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCFOFFSETCNT	R	0h	<p>Blanking Offset Counter</p> <p>These 16-bits are read only and indicate the current value of the offset counter. The counter counts down to zero and then stops until it is re-loaded on the next period or zero event as defined by the DCFCTL[PULSESEL] bit. The offset counter is not affected by the free/soft emulation bits. That is, it will always continue to count down if the device is halted by a emulation stop.</p> <p>Reset type: SYSRSn</p>



**18.16.2.72 DCFWINDOW Register (Offset = CBh) [reset = 0h]**

DCFWINDOW is shown in [Figure 18-165](#) and described in [Table 18-95](#).

Return to the [Summary Table](#).

Digital Compare Filter Window Register

**Figure 18-165. DCFWINDOW Register**

15	14	13	12	11	10	9	8
DCFWINDOW							
R/W-0h							
7	6	5	4	3	2	1	0
DCFWINDOW							
R/W-0h							

**Table 18-95. DCFWINDOW Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCFWINDOW	R/W	0h	<p>Blanking Window Width</p> <p>00h: No blanking window is generated.</p> <p>01-FFFFh: Specifies the width of the blanking window in TBCLK cycles. The blanking window begins when the offset counter expires. When this occurs, the window counter is loaded and begins to count down. If the blanking window is currently active and the offset counter expires, the blanking window counter is not restarted and the blanking window is cut short prematurely. Care should be taken to avoid this situation. The blanking window can cross a PWM period boundary.</p> <p>Reset type: SYSRSn</p>

**18.16.2.73 DCFWINDOWCNT Register (Offset = CCh) [reset = 0h]**

DCFWINDOWCNT is shown in [Figure 18-166](#) and described in [Table 18-96](#).

Return to the [Summary Table](#).

Digital Compare Filter Window Counter Register

**Figure 18-166. DCFWINDOWCNT Register**

15	14	13	12	11	10	9	8
DCFWINDOWCNT							
R-0h							
7	6	5	4	3	2	1	0
DCFWINDOWCNT							
R-0h							

**Table 18-96. DCFWINDOWCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCFWINDOWCNT	R	0h	Blanking Window Counter  These 16 bits are read only and indicate the current value of the window counter. The counter counts down to zero and then stops until it is re-loaded when the offset counter reaches zero again. Reset type: SYSRSn

**18.16.2.74 DCCAP Register (Offset = CFh) [reset = 0h]**

DCCAP is shown in [Figure 18-167](#) and described in [Table 18-97](#).

Return to the [Summary Table](#).

Digital Compare Counter Capture Register

**Figure 18-167. DCCAP Register**

15	14	13	12	11	10	9	8
DCCAP							
R-0h							
7	6	5	4	3	2	1	0
DCCAP							
R-0h							

**Table 18-97. DCCAP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCCAP	R	0h	<p>Digital Compare Time-Base Counter Capture</p> <p>To enable time-base counter capture, set the DCCAPCLT[CAPE] bit to 1. If enabled, reflects the value of the time-base counter (TBCTR) on the low to high edge transition of a filtered (DCEVTFLT) event. Further capture events are ignored until the next period or zero as selected by the DCFCTL[PULSESEL] bit. Shadowing of DCCAP is enabled and disabled by the DCCAPCTL[SHDWMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>- If DCCAPCTL[SHDWMODE] = 0, then the shadow is enabled. In this mode, the active register is copied to the shadow register on the TBCTR = TBPRD or TBCTR = zero as defined by the DCFCTL[PULSESEL] bit. CPU reads of this register will return the shadow register value.</li> <li>- If DCCAPCTL[SHDWMODE] = 1, then the shadow register is disabled. In this mode, CPU reads will return the active register value. The active and shadow registers share the same memory map address.</li> </ul> <p>Reset type: SYSRSn</p>

**18.16.2.75 DCAHTRIPSEL Register (Offset = D2h) [reset = 0h]**

DCAHTRIPSEL is shown in [Figure 18-168](#) and described in [Table 18-98](#).

Return to the [Summary Table](#).

Digital Compare AH Trip Select

**Figure 18-168. DCAHTRIPSEL Register**

15		14		13		12		11		10		9		8	
RESERVED	TRIPINPUT15	TRIPINPUT14	RESERVED	TRIPINPUT12	TRIPINPUT11	TRIPINPUT10	TRIPINPUT9								
R-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	
TRIPINPUT8	TRIPINPUT7	TRIPINPUT6	TRIPINPUT5	TRIPINPUT4	TRIPINPUT3	TRIPINPUT2	TRIPINPUT1								
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 18-98. DCAHTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	TRIPINPUT15	R/W	0h	TRIP Input 15 0: Trip Input 15 not selected as combinational ORed input 1: Trip Input 15 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
13	TRIPINPUT14	R/W	0h	TRIP Input 14 0: Trip Input 14 not selected as combinational ORed input 1: Trip Input 14 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	TRIPINPUT12	R/W	0h	TRIP Input 12 0: Trip Input 12 not selected as combinational ORed input 1: Trip Input 12 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
10	TRIPINPUT11	R/W	0h	TRIP Input 11 0: Trip Input 11 not selected as combinational ORed input 1: Trip Input 11 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
9	TRIPINPUT10	R/W	0h	TRIP Input 10 0: Trip Input 10 not selected as combinational ORed input 1: Trip Input 10 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
8	TRIPINPUT9	R/W	0h	TRIP Input 9 0: Trip Input 9 not selected as combinational ORed input 1: Trip Input 9 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
7	TRIPINPUT8	R/W	0h	TRIP Input 8 0: Trip Input 8 not selected as combinational ORed input 1: Trip Input 8 selected as combinational ORed input to DCAH mux Reset type: SYSRSn

**Table 18-98. DCAHTRIPSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	TRIPINPUT7	R/W	0h	TRIP Input 7 0: Trip Input 7 not selected as combinational ORed input 1: Trip Input 7 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
5	TRIPINPUT6	R/W	0h	TRIP Input 6 0: Trip Input 6 not selected as combinational ORed input 1: Trip Input 6 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
4	TRIPINPUT5	R/W	0h	TRIP Input 5 0: Trip Input 5 not selected as combinational ORed input 1: Trip Input 5 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
3	TRIPINPUT4	R/W	0h	TRIP Input 4 0: Trip Input 4 not selected as combinational ORed input 1: Trip Input 4 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
2	TRIPINPUT3	R/W	0h	TRIP Input 3 0: Trip Input 3 not selected as combinational ORed input 1: Trip Input 3 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
1	TRIPINPUT2	R/W	0h	TRIP Input 2 0: Trip Input 2 not selected as combinational ORed input 1: Trip Input 2 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
0	TRIPINPUT1	R/W	0h	TRIP Input 1 0: Trip Input 1 not selected as combinational ORed input 1: Trip Input 1 selected as combinational ORed input to DCAH mux Reset type: SYSRSn

**18.16.2.76 DCALTRIPSEL Register (Offset = D3h) [reset = 0h]**

DCALTRIPSEL is shown in [Figure 18-169](#) and described in [Table 18-99](#).

Return to the [Summary Table](#).

Digital Compare AL Trip Select

**Figure 18-169. DCALTRIPSEL Register**

15		14		13		12		11		10		9		8	
RESERVED	TRIPINPUT15	TRIPINPUT14	RESERVED	TRIPINPUT12	TRIPINPUT11	TRIPINPUT10	TRIPINPUT9								
R-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	
TRIPINPUT8	TRIPINPUT7	TRIPINPUT6	TRIPINPUT5	TRIPINPUT4	TRIPINPUT3	TRIPINPUT2	TRIPINPUT1								
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 18-99. DCALTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	TRIPINPUT15	R/W	0h	TRIP Input 15 0: Trip Input 15 not selected as combinational ORed input 1: Trip Input 15 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
13	TRIPINPUT14	R/W	0h	TRIP Input 14 0: Trip Input 14 not selected as combinational ORed input 1: Trip Input 14 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	TRIPINPUT12	R/W	0h	TRIP Input 12 0: Trip Input 12 not selected as combinational ORed input 1: Trip Input 12 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
10	TRIPINPUT11	R/W	0h	TRIP Input 11 0: Trip Input 11 not selected as combinational ORed input 1: Trip Input 11 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
9	TRIPINPUT10	R/W	0h	TRIP Input 10 0: Trip Input 10 not selected as combinational ORed input 1: Trip Input 10 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
8	TRIPINPUT9	R/W	0h	TRIP Input 9 0: Trip Input 9 not selected as combinational ORed input 1: Trip Input 9 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
7	TRIPINPUT8	R/W	0h	TRIP Input 8 0: Trip Input 8 not selected as combinational ORed input 1: Trip Input 8 selected as combinational ORed input to DCAL mux Reset type: SYSRSn

**Table 18-99. DCALTRIPSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	TRIPINPUT7	R/W	0h	TRIP Input 7 0: Trip Input 7 not selected as combinational ORed input 1: Trip Input 7 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
5	TRIPINPUT6	R/W	0h	TRIP Input 6 0: Trip Input 6 not selected as combinational ORed input 1: Trip Input 6 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
4	TRIPINPUT5	R/W	0h	TRIP Input 5 0: Trip Input 5 not selected as combinational ORed input 1: Trip Input 5 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
3	TRIPINPUT4	R/W	0h	TRIP Input 4 0: Trip Input 4 not selected as combinational ORed input 1: Trip Input 4 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
2	TRIPINPUT3	R/W	0h	TRIP Input 3 0: Trip Input 3 not selected as combinational ORed input 1: Trip Input 3 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
1	TRIPINPUT2	R/W	0h	TRIP Input 2 0: Trip Input 2 not selected as combinational ORed input 1: Trip Input 2 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
0	TRIPINPUT1	R/W	0h	TRIP Input 1 0: Trip Input 1 not selected as combinational ORed input 1: Trip Input 1 selected as combinational ORed input to DCAL mux Reset type: SYSRSn

**18.16.2.77 DCBHTRIPSEL Register (Offset = D4h) [reset = 0h]**

DCBHTRIPSEL is shown in [Figure 18-170](#) and described in [Table 18-100](#).

Return to the [Summary Table](#).

Digital Compare BH Trip Select

**Figure 18-170. DCBHTRIPSEL Register**

15		14		13		12		11		10		9		8	
RESERVED	TRIPINPUT15	TRIPINPUT14	RESERVED	TRIPINPUT12	TRIPINPUT11	TRIPINPUT10	TRIPINPUT9								
R-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	
TRIPINPUT8	TRIPINPUT7	TRIPINPUT6	TRIPINPUT5	TRIPINPUT4	TRIPINPUT3	TRIPINPUT2	TRIPINPUT1								
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 18-100. DCBHTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	TRIPINPUT15	R/W	0h	TRIP Input 15 0: Trip Input 15 not selected as combinational ORed input 1: Trip Input 15 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
13	TRIPINPUT14	R/W	0h	TRIP Input 14 0: Trip Input 14 not selected as combinational ORed input 1: Trip Input 14 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	TRIPINPUT12	R/W	0h	TRIP Input 12 0: Trip Input 12 not selected as combinational ORed input 1: Trip Input 12 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
10	TRIPINPUT11	R/W	0h	TRIP Input 11 0: Trip Input 11 not selected as combinational ORed input 1: Trip Input 11 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
9	TRIPINPUT10	R/W	0h	TRIP Input 10 0: Trip Input 10 not selected as combinational ORed input 1: Trip Input 10 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
8	TRIPINPUT9	R/W	0h	TRIP Input 9 0: Trip Input 9 not selected as combinational ORed input 1: Trip Input 9 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
7	TRIPINPUT8	R/W	0h	TRIP Input 8 0: Trip Input 8 not selected as combinational ORed input 1: Trip Input 8 selected as combinational ORed input to DCBH mux Reset type: SYSRSn



**Table 18-100. DCBHTRIPSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	TRIPINPUT7	R/W	0h	TRIP Input 7 0: Trip Input 7 not selected as combinational ORed input 1: Trip Input 7 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
5	TRIPINPUT6	R/W	0h	TRIP Input 6 0: Trip Input 6 not selected as combinational ORed input 1: Trip Input 6 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
4	TRIPINPUT5	R/W	0h	TRIP Input 5 0: Trip Input 5 not selected as combinational ORed input 1: Trip Input 5 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
3	TRIPINPUT4	R/W	0h	TRIP Input 4 0: Trip Input 4 not selected as combinational ORed input 1: Trip Input 4 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
2	TRIPINPUT3	R/W	0h	TRIP Input 3 0: Trip Input 3 not selected as combinational ORed input 1: Trip Input 3 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
1	TRIPINPUT2	R/W	0h	TRIP Input 2 0: Trip Input 2 not selected as combinational ORed input 1: Trip Input 2 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
0	TRIPINPUT1	R/W	0h	TRIP Input 1 0: Trip Input 1 not selected as combinational ORed input 1: Trip Input 1 selected as combinational ORed input to DCBH mux Reset type: SYSRSn

**18.16.2.78 DCBLTRIPSEL Register (Offset = D5h) [reset = 0h]**

DCBLTRIPSEL is shown in [Figure 18-171](#) and described in [Table 18-101](#).

Return to the [Summary Table](#).

Digital Compare BL Trip Select

**Figure 18-171. DCBLTRIPSEL Register**

15		14		13		12		11		10		9		8	
RESERVED	TRIPINPUT15	TRIPINPUT14	RESERVED	TRIPINPUT12	TRIPINPUT11	TRIPINPUT10	TRIPINPUT9								
R-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	
TRIPINPUT8	TRIPINPUT7	TRIPINPUT6	TRIPINPUT5	TRIPINPUT4	TRIPINPUT3	TRIPINPUT2	TRIPINPUT1								
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 18-101. DCBLTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	TRIPINPUT15	R/W	0h	TRIP Input 15 0: Trip Input 15 not selected as combinational ORed input 1: Trip Input 15 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
13	TRIPINPUT14	R/W	0h	TRIP Input 14 0: Trip Input 14 not selected as combinational ORed input 1: Trip Input 14 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	TRIPINPUT12	R/W	0h	TRIP Input 12 0: Trip Input 12 not selected as combinational ORed input 1: Trip Input 12 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
10	TRIPINPUT11	R/W	0h	TRIP Input 11 0: Trip Input 11 not selected as combinational ORed input 1: Trip Input 11 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
9	TRIPINPUT10	R/W	0h	TRIP Input 10 0: Trip Input 10 not selected as combinational ORed input 1: Trip Input 10 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
8	TRIPINPUT9	R/W	0h	TRIP Input 9 0: Trip Input 9 not selected as combinational ORed input 1: Trip Input 9 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
7	TRIPINPUT8	R/W	0h	TRIP Input 8 0: Trip Input 8 not selected as combinational ORed input 1: Trip Input 8 selected as combinational ORed input to DCAL mux Reset type: SYSRSn

**Table 18-101. DCBLTRIPSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	TRIPINPUT7	R/W	0h	TRIP Input 7 0: Trip Input 7 not selected as combinational ORed input 1: Trip Input 7 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
5	TRIPINPUT6	R/W	0h	TRIP Input 6 0: Trip Input 6 not selected as combinational ORed input 1: Trip Input 6 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
4	TRIPINPUT5	R/W	0h	TRIP Input 5 0: Trip Input 5 not selected as combinational ORed input 1: Trip Input 5 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
3	TRIPINPUT4	R/W	0h	TRIP Input 4 0: Trip Input 4 not selected as combinational ORed input 1: Trip Input 4 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
2	TRIPINPUT3	R/W	0h	TRIP Input 3 0: Trip Input 3 not selected as combinational ORed input 1: Trip Input 3 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
1	TRIPINPUT2	R/W	0h	TRIP Input 2 0: Trip Input 2 not selected as combinational ORed input 1: Trip Input 2 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
0	TRIPINPUT1	R/W	0h	TRIP Input 1 0: Trip Input 1 not selected as combinational ORed input 1: Trip Input 1 selected as combinational ORed input to DCAL mux Reset type: SYSRSn

**18.16.2.79 EPWMLOCK Register (Offset = FAh) [reset = 0h]**

 EPWMLOCK is shown in [Figure 18-172](#) and described in [Table 18-102](#).

 Return to the [Summary Table](#).

EPWM Lock Register

**Figure 18-172. EPWMLOCK Register**

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

**Table 18-102. EPWMLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	Write to this register succeeds only if this field is written with a value of 0xa5a5  Note: [1] Due to this KEY, only 32-bit writes will succeed (provided the KEY matches). 16-bit writes to the upper or lower half of this register will be ignored Reset type: SYSRSn
15-5	RESERVED	R	0h	Reserved
4	DCLOCK	R/WOnce	0h	0: Digital Compare registers from 0xC0 to 0xD5 offsets are protected by EALLOW. 1: Digital Compare registers from 0xC0 and 0xD5 offsets are locked and not writable. Reset type: SYSRSn
3	TZCLRLOCK	R/WOnce	0h	0: Digital Compare registers from 0x97 to 0x9B offsets are protected by EALLOW. 1: Digital Compare registers from 0x97 and 0x9B offsets are locked and not writable. Reset type: SYSRSn
2	TZCFGLOCK	R/WOnce	0h	0: TripZone registers from 0x80 to 0x8D offsets are protected by EALLOW. 1: TripZone registers from 0x80 and 0x8D offsets are locked and not writable. Reset type: SYSRSn
1	GLLOCK	R/WOnce	0h	0: TripZone registers from 0x34 to 0x35 offsets are protected by EALLOW. 1: TripZone registers from 0x34 to 0x35 offsets are locked and not writable Reset type: SYSRSn

**Table 18-102. EPWMLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	HRLOCK	R/WOnce	0h	0: HRPWM registers from 0x20 to 0x2D offsets are protected by EALLOW 1: HRPWM registers from 0x20 and 0x2D offsets are locked and not writable. Reset type: SYSRSn

**18.16.2.80 HWVDELVAL Register (Offset = FDh) [reset = 0h]**

HWVDELVAL is shown in [Figure 18-173](#) and described in [Table 18-103](#).

Return to the [Summary Table](#).

Hardware Valley Mode Delay Register

**Figure 18-173. HWVDELVAL Register**

15	14	13	12	11	10	9	8
HWVDELVAL							
R-0h							
7	6	5	4	3	2	1	0
HWVDELVAL							
R-0h							

**Table 18-103. HWVDELVAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	HWVDELVAL	R	0h	<p>Hardware Valley Delay Value Register</p> <p>This read only register reflects the hardware delay value calculated by the equations defined in VCAPCTL[VDELAYDIV]. This reflects the latest value from the hardware calculations and can change every time valley capture sequence is triggered and VCAP1 and VCAP2 values are updated.</p> <p>Reset type: SYSRSn</p>

**18.16.2.81 VCNTVAL Register (Offset = FEh) [reset = 0h]**

VCNTVAL is shown in [Figure 18-174](#) and described in [Table 18-104](#).

Return to the [Summary Table](#).

Hardware Valley Counter Register

**Figure 18-174. VCNTVAL Register**

15	14	13	12	11	10	9	8
VCNTVAL							
R-0h							
7	6	5	4	3	2	1	0
VCNTVAL							
R-0h							

**Table 18-104. VCNTVAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	VCNTVAL	R	0h	Valley Time Base Counter Register  This register reflects the captured VCNT value upon occurrence of STOPENGE selected in VCNTCFG register. Reset type: SYSRSn

### 18.16.3 SYNC\_SOC\_REGS Registers

Table 18-105 lists the SYNC\_SOC\_REGS registers. All register offset addresses not listed in Table 18-105 should be considered as reserved locations and the register contents should not be modified.

**Table 18-105. SYNC\_SOC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	SYNCSELECT	Sync Input and Output Select Register	EALLOW	<a href="#">Go</a>
2h	ADCSOCOUTSELECT	External ADCSOC Select Register	EALLOW	<a href="#">Go</a>
4h	SYNCSOCLOCK	SYNCSEL and EXTADCSOC Select Lock register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 18-106 shows the codes that are used for access types in this section.

**Table 18-106. SYNC\_SOC\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
WOnce	WOnce	Write Set once
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.



### 18.16.3.1 SYNCSELECT Register (Offset = 0h) [reset = E003FFFFh]

SYNCSELECT is shown in [Figure 18-175](#) and described in [Table 18-107](#).

Return to the [Summary Table](#).

Sync Input and Output Select Register

**Figure 18-175. SYNCSELECT Register**

31	30	29	28	27	26	25	24
EPWM1SYNCIN			SYNCOUT		RESERVED		
R/W-7h			R/W-0h		R-0-0h		
23	22	21	20	19	18	17	16
RESERVED						ECAP6SYNCIN	
R-0-0h						R/W-7h	
15	14	13	12	11	10	9	8
ECAP6SYNCIN	ECAP4SYNCIN			ECAP1SYNCIN			RESERVED
R/W-7h	R/W-7h			R/W-7h			
7	6	5	4	3	2	1	0
RESERVED		EPWM7SYNCIN			EPWM4SYNCIN		
		R/W-7h			R/W-7h		

**Table 18-107. SYNCSELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	EPWM1SYNCIN	R/W	7h	<p>Selects Sync Input Source for EPWM1:</p> <p>000: EXTSYNCIN1 selected            001: Reserved            010: Reserved            011: Reserved            100: Reserved            101: Reserved            110: Reserved            111: Reserved</p> <p>Notes:            [1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.            Reset type: SYSRSn</p>
28-27	SYNCOUT	R/W	0h	<p>Select Syncout Source:</p> <p>00: EPWM1SYNCOUT selected            01: EPWM4SYNCOUT selected            10: EPWM7SYNCOUT selected            11: EPWM10SYNCOUT selected(Reserved)</p> <p>Notes:            [1] Reserved position defaults to 00 selection            Reset type: SYSRSn</p>
26-18	RESERVED	R-0	0h	Reserved

**Table 18-107. SYNCSELECT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17-15	ECAP6SYNCIN	R/W	7h	<p>Selects Sync Input Source for ECAP4:</p> <p>000: EPWM1SYNCOOUT selected  001: EPWM4SYNCOOUT selected  010: EPWM7SYNCOOUT selected  011: EPWM10SYNCOOUT selected(Reserved)  100: ECAP1SYNCOOUT selected  101: EXTSYNCIN1 selected  110: EXTSYNCIN2 selected  111: ECAP4SYNCOOUT selected</p> <p>Notes:  [1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.  Reset type: SYSRSn</p>
14-12	ECAP4SYNCIN	R/W	7h	<p>Selects Sync Input Source for ECAP4:</p> <p>000: EPWM1SYNCOOUT selected  001: EPWM4SYNCOOUT selected  010: EPWM7SYNCOOUT selected  011: EPWM10SYNCOOUT selected(Reserved)  100: ECAP1SYNCOOUT selected  101: EXTSYNCIN1 selected  110: EXTSYNCIN2 selected  111: ECAP4SYNCOOUT selected (Reserved)</p> <p>Notes:  [1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.  Reset type: SYSRSn</p>
11-9	ECAP1SYNCIN	R/W	7h	<p>Selects Sync Input Source for ECAP1:</p> <p>000: EPWM1SYNCOOUT selected  001: EPWM4SYNCOOUT selected  010: EPWM7SYNCOOUT selected  011: EPWM10SYNCOOUT selected(Reserved)  100: ECAP1SYNCOOUT selected (Reserved)  101: EXTSYNCIN1 selected  110: EXTSYNCIN2 selected  111: ECAP4SYNCOOUT selected (Reserved)</p> <p>Notes:  [1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.  Reset type: SYSRSn</p>
8-6	RESERVED	R/W	7h	Reserved

**Table 18-107. SYNCSELECT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-3	EPWM7SYNCIN	R/W	7h	Selects Sync Input Source for EPWM7: 000: EPWM1SYNCOOUT selected 001: EPWM4SYNCOOUT selected 010: EPWM7SYNCOOUT selected (Reserved) 011: EPWM10SYNCOOUT selected (Reserved) 100: ECAP1SYNCOOUT selected (Reserved) 101: EXTSYNCCIN1 selected 110: EXTSYNCCIN2 selected 111: ECAP4SYNCOOUT selected (Reserved)  Notes: [1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low. Reset type: SYSRSn
2-0	EPWM4SYNCIN	R/W	7h	Selects Sync Input Source for EPWM4: 000: EPWM1SYNCOOUT selected 001: EPWM4SYNCOOUT selected (Reserved) 010: EPWM7SYNCOOUT selected (Reserved) 011: EPWM10SYNCOOUT selected (Reserved) 100: ECAP1SYNCOOUT selected (Reserved) 101: EXTSYNCCIN1 selected 110: EXTSYNCCIN2 selected 111: Reserved  Notes: [1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low. Reset type: SYSRSn

**18.16.3.2 ADCSOCOUTSELECT Register (Offset = 2h) [reset = 0h]**

 ADCSOCOUTSELECT is shown in [Figure 18-176](#) and described in [Table 18-108](#).

 Return to the [Summary Table](#).

External ADCSOC Select Register

**Figure 18-176. ADCSOCOUTSELECT Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED	RESERVED	RESERVED	RESERVED
R-0-0h							
23	22	21	20	19	18	17	16
PWM8SOCBE N	PWM7SOCBE N	PWM6SOCBE N	PWM5SOCBE N	PWM4SOCBE N	PWM3SOCBE N	PWM2SOCBE N	PWM1SOCBE N
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				RESERVED	RESERVED	RESERVED	RESERVED
R-0-0h							
7	6	5	4	3	2	1	0
PWM8SOCAE N	PWM7SOCAE N	PWM6SOCAE N	PWM5SOCAE N	PWM4SOCAE N	PWM3SOCAE N	PWM2SOCAE N	PWM1SOCAE N
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 18-108. ADCSOCOUTSELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R-0	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	PWM8SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
22	PWM7SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
21	PWM6SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
20	PWM5SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
19	PWM4SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn

**Table 18-108. ADCSOCOUTSELECT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	PWM3SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
17	PWM2SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
16	PWM1SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
15-12	RESERVED	R-0	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	PWM8SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
6	PWM7SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
5	PWM6SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
4	PWM5SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
3	PWM4SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
2	PWM3SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn

**Table 18-108. ADCSOCOUTSELECT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	PWM2SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
0	PWM1SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn

### 18.16.3.3 SYNCLOCK Register (Offset = 4h) [reset = 0h]

SYNCLOCK is shown in [Figure 18-177](#) and described in [Table 18-109](#).

Return to the [Summary Table](#).

SYNCSEL and EXTADCSOC Select Lock register

**Figure 18-177. SYNCLOCK Register**

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

**Table 18-109. SYNCLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	ADCSOCOUTSELECT	R/WOnce	0h	ADCSOCOUTSELECT Register Lock bit: 0: Respective register is not locked 1: Respective register is locked. Notes: [1] Any bit in this register, once set can only be created through a SYSRSn. Write of 0 to any bit of this register has no effect [2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed Reset type: SYSRSn
0	SYNCSELECT	R/WOnce	0h	SYNCSELECT Register Lock bit: 0: Respective register is not locked 1: Respective register is locked. Notes: [1] Any bit in this register, once set can only be created through a SYSRSn. Write of 0 to any bit of this register has no effect [2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed Reset type: SYSRSn

## 18.16.4 Register to Driverlib Function Mapping

**Table 18-110. EPWM Registers to Driverlib Functions**

File	Driverlib Function
<b>TBCTL</b>	
epwm.c	EPWM_setEmulationMode
epwm.h	EPWM_setCountModeAfterSync
epwm.h	EPWM_setClockPrescaler
epwm.h	EPWM_forceSyncPulse
epwm.h	EPWM_setSyncOutPulseMode
epwm.h	EPWM_setPeriodLoadMode
epwm.h	EPWM_enablePhaseShiftLoad
epwm.h	EPWM_disablePhaseShiftLoad
epwm.h	EPWM_setTimeBaseCounterMode
epwm.h	EPWM_selectPeriodLoadEvent
epwm.h	EPWM_enableOneShotSync
epwm.h	EPWM_disableOneShotSync
epwm.h	EPWM_startOneShotSync
<b>TBCTL2</b>	
epwm.h	EPWM_setSyncOutPulseMode
epwm.h	EPWM_selectPeriodLoadEvent
epwm.h	EPWM_enableOneShotSync
epwm.h	EPWM_disableOneShotSync
epwm.h	EPWM_startOneShotSync
<b>TBCTR</b>	
epwm.h	EPWM_setTimeBaseCounter
epwm.h	EPWM_getTimeBaseCounterValue
<b>TBSTS</b>	
epwm.h	EPWM_getTimeBaseCounterOverflowStatus
epwm.h	EPWM_clearTimeBaseCounterOverflowEvent
epwm.h	EPWM_getSyncStatus
epwm.h	EPWM_clearSyncEvent
epwm.h	EPWM_getTimeBaseCounterDirection
<b>CMPCTL</b>	
epwm.h	EPWM_setCounterCompareShadowLoadMode
epwm.h	EPWM_disableCounterCompareShadowLoadMode
epwm.h	EPWM_getCounterCompareShadowStatus
<b>CMPCTL2</b>	
epwm.h	EPWM_setCounterCompareShadowLoadMode
epwm.h	EPWM_disableCounterCompareShadowLoadMode
<b>DBCTL</b>	
epwm.h	EPWM_setDeadBandOutputSwapMode
epwm.h	EPWM_setDeadBandDelayMode
epwm.h	EPWM_setDeadBandDelayPolarity
epwm.h	EPWM_setRisingEdgeDeadBandDelayInput
epwm.h	EPWM_setFallingEdgeDeadBandDelayInput
epwm.h	EPWM_setDeadBandControlShadowLoadMode
epwm.h	EPWM_disableDeadBandControlShadowLoadMode
epwm.h	EPWM_setRisingEdgeDelayCountShadowLoadMode
epwm.h	EPWM_disableRisingEdgeDelayCountShadowLoadMode



**Table 18-110. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
epwm.h	EPWM_setFallingEdgeDelayCountShadowLoadMode
epwm.h	EPWM_disableFallingEdgeDelayCountShadowLoadMode
epwm.h	EPWM_setDeadBandCounterClock
<b>DBCTL2</b>	
epwm.h	EPWM_setDeadBandControlShadowLoadMode
epwm.h	EPWM_disableDeadBandControlShadowLoadMode
<b>AQCTL</b>	
epwm.h	EPWM_setActionQualifierShadowLoadMode
epwm.h	EPWM_disableActionQualifierShadowLoadMode
epwm.h	EPWM_setActionQualifierAction
epwm.h	EPWM_setActionQualifierActionComplete
epwm.h	EPWM_setAdditionalActionQualifierActionComplete
<b>AQTSRCSEL</b>	
epwm.h	EPWM_setActionQualifierT1TriggerSource
epwm.h	EPWM_setActionQualifierT2TriggerSource
<b>PCCTL</b>	
epwm.h	EPWM_enableChopper
epwm.h	EPWM_disableChopper
epwm.h	EPWM_setChopperDutyCycle
epwm.h	EPWM_setChopperFreq
epwm.h	EPWM_setChopperFirstPulseWidth
<b>VCAPCTL</b>	
epwm.h	EPWM_enableValleyCapture
epwm.h	EPWM_disableValleyCapture
epwm.h	EPWM_startValleyCapture
epwm.h	EPWM_setValleyTriggerSource
epwm.h	EPWM_enableValleyHWDelay
epwm.h	EPWM_disableValleyHWDelay
epwm.h	EPWM_setValleyDelayDivider
<b>VCNTCFG</b>	
epwm.h	EPWM_setValleyTriggerEdgeCounts
epwm.h	EPWM_getValleyEdgeStatus
<b>GLDCTL</b>	
epwm.h	EPWM_enableGlobalLoad
epwm.h	EPWM_disableGlobalLoad
epwm.h	EPWM_setGlobalLoadTrigger
epwm.h	EPWM_setGlobalLoadEventPrescale
epwm.h	EPWM_getGlobalLoadEventCount
epwm.h	EPWM_disableGlobalLoadOneShotMode
epwm.h	EPWM_enableGlobalLoadOneShotMode
epwm.h	EPWM_setGlobalLoadOneShotLatch
epwm.h	EPWM_forceGlobalLoadOneShotEvent
<b>GLDCFG</b>	
epwm.h	EPWM_enableGlobalLoadRegisters
epwm.h	EPWM_disableGlobalLoadRegisters
<b>XLINK</b>	
epwm.h	EPWM_setupEPWMLinks

**Table 18-110. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>AQCTLA</b>	
epwm.h	EPWM_setActionQualifierAction
epwm.h	EPWM_setActionQualifierActionComplete
epwm.h	EPWM_setAdditionalActionQualifierActionComplete
<b>AQCTLA2</b>	
epwm.h	EPWM_setActionQualifierAction
epwm.h	EPWM_setAdditionalActionQualifierActionComplete
<b>AQCTLB</b>	
-	See AQCTLA
<b>AQCTLB2</b>	
-	See AQCTLA2
<b>AQSFRC</b>	
epwm.h	EPWM_setActionQualifierContSWForceShadowMode
epwm.h	EPWM_setActionQualifierSWAction
epwm.h	EPWM_forceActionQualifierSWAction
<b>AQCSFRC</b>	
epwm.h	EPWM_setActionQualifierContSWForceAction
<b>DBRED</b>	
epwm.h	EPWM_setRisingEdgeDelayCount
<b>DBFED</b>	
epwm.h	EPWM_setFallingEdgeDelayCount
<b>TBPHS</b>	
epwm.h	EPWM_setPhaseShift
<b>TBPRD</b>	
epwm.h	EPWM_setTimeBasePeriod
epwm.h	EPWM_getTimeBasePeriod
<b>CMPA</b>	
epwm.h	EPWM_setCounterCompareValue
epwm.h	EPWM_getCounterCompareValue
<b>CMPB</b>	
-	See CMPA
<b>CMPC</b>	
epwm.h	EPWM_setCounterCompareShadowLoadMode
epwm.h	EPWM_disableCounterCompareShadowLoadMode
epwm.h	EPWM_getCounterCompareShadowStatus
<b>CMPD</b>	
-	See CMPC
<b>GLDCTL2</b>	
epwm.h	EPWM_setGlobalLoadOneShotLatch
epwm.h	EPWM_forceGlobalLoadOneShotEvent
<b>SWVDELVAL</b>	
epwm.h	EPWM_setValleySWDelayValue
<b>TZSEL</b>	
epwm.h	EPWM_enableTripZoneSignals
epwm.h	EPWM_disableTripZoneSignals
<b>TZDCSEL</b>	
epwm.h	EPWM_setTripZoneDigitalCompareEventCondition

**Table 18-110. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>TZCTL</b>	
epwm.h	EPWM_enableTripZoneAdvAction
epwm.h	EPWM_disableTripZoneAdvAction
epwm.h	EPWM_setTripZoneAction
epwm.h	EPWM_setTripZoneAdvAction
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionA
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionB
<b>TZCTL2</b>	
epwm.h	EPWM_enableTripZoneAdvAction
epwm.h	EPWM_disableTripZoneAdvAction
epwm.h	EPWM_setTripZoneAdvAction
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionA
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionB
<b>TZCTLDCA</b>	
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionA
<b>TZCTLDCB</b>	
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionB
<b>TZEINT</b>	
epwm.h	EPWM_enableTripZoneInterrupt
epwm.h	EPWM_disableTripZoneInterrupt
<b>TZFLG</b>	
epwm.h	EPWM_getTripZoneFlagStatus
<b>TZCBCFLG</b>	
epwm.h	EPWM_getCycleByCycleTripZoneFlagStatus
<b>TZOSTFLG</b>	
epwm.h	EPWM_getOneShotTripZoneFlagStatus
<b>TZCLR</b>	
epwm.h	EPWM_selectCycleByCycleTripZoneClearEvent
epwm.h	EPWM_clearTripZoneFlag
<b>TZCBCCLR</b>	
epwm.h	EPWM_clearCycleByCycleTripZoneFlag
<b>TZOSTCLR</b>	
epwm.h	EPWM_clearOneShotTripZoneFlag
<b>TZFRC</b>	
epwm.h	EPWM_forceTripZoneEvent
<b>ETSEL</b>	
epwm.h	EPWM_enableInterrupt
epwm.h	EPWM_disableInterrupt
epwm.h	EPWM_setInterruptSource
epwm.h	EPWM_enableADCTrigger
epwm.h	EPWM_disableADCTrigger
epwm.h	EPWM_setADCTriggerSource
<b>ETPS</b>	
epwm.h	EPWM_setInterruptEventCount
epwm.h	EPWM_setADCTriggerEventPrescale
<b>ETFLG</b>	
epwm.h	EPWM_getEventTriggerInterruptStatus

**Table 18-110. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
epwm.h	EPWM_getADCTriggerFlagStatus
<b>ETCLR</b>	
epwm.h	EPWM_clearEventTriggerInterruptFlag
epwm.h	EPWM_clearADCTriggerFlag
<b>ETFRC</b>	
epwm.h	EPWM_forceEventTriggerInterrupt
epwm.h	EPWM_forceADCTrigger
<b>ETINTPS</b>	
epwm.h	EPWM_setInterruptEventCount
epwm.h	EPWM_getInterruptEventCount
<b>ETSOCP</b>	
epwm.h	EPWM_setADCTriggerEventPrescale
epwm.h	EPWM_getADCTriggerEventCount
<b>ETCNTINITCTL</b>	
epwm.h	EPWM_enableInterruptEventCountInit
epwm.h	EPWM_disableInterruptEventCountInit
epwm.h	EPWM_forceInterruptEventCountInit
epwm.h	EPWM_enableADCTriggerEventCountInit
epwm.h	EPWM_disableADCTriggerEventCountInit
epwm.h	EPWM_forceADCTriggerEventCountInit
<b>ETCNTINIT</b>	
epwm.h	EPWM_enableInterruptEventCountInit
epwm.h	EPWM_disableInterruptEventCountInit
epwm.h	EPWM_forceInterruptEventCountInit
epwm.h	EPWM_setInterruptEventCountInitValue
epwm.h	EPWM_enableADCTriggerEventCountInit
epwm.h	EPWM_disableADCTriggerEventCountInit
epwm.h	EPWM_forceADCTriggerEventCountInit
epwm.h	EPWM_setADCTriggerEventCountInitValue
<b>DCTRISEL</b>	
epwm.h	EPWM_selectDigitalCompareTripInput
epwm.h	EPWM_enableDigitalCompareTripCombinationInput
<b>DCACTL</b>	
epwm.h	EPWM_setDigitalCompareEventSource
epwm.h	EPWM_setDigitalCompareEventSyncMode
epwm.h	EPWM_enableDigitalCompareADCTrigger
epwm.h	EPWM_disableDigitalCompareADCTrigger
epwm.h	EPWM_enableDigitalCompareSyncEvent
epwm.h	EPWM_disableDigitalCompareSyncEvent
<b>DCBCTL</b>	
-	See DCACTL
<b>DCFCTL</b>	
epwm.h	EPWM_enableDigitalCompareBlankingWindow
epwm.h	EPWM_disableDigitalCompareBlankingWindow
epwm.h	EPWM_enableDigitalCompareWindowInverseMode
epwm.h	EPWM_disableDigitalCompareWindowInverseMode
epwm.h	EPWM_setDigitalCompareBlankingEvent

**Table 18-110. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
epwm.h	EPWM_setDigitalCompareFilterInput
epwm.h	EPWM_enableDigitalCompareEdgeFilter
epwm.h	EPWM_disableDigitalCompareEdgeFilter
epwm.h	EPWM_setDigitalCompareEdgeFilterMode
epwm.h	EPWM_setDigitalCompareEdgeFilterEdgeCount
epwm.h	EPWM_getDigitalCompareEdgeFilterEdgeCount
epwm.h	EPWM_getDigitalCompareEdgeFilterEdgeStatus
<b>DCCAPCTL</b>	
epwm.h	EPWM_enableDigitalCompareCounterCapture
epwm.h	EPWM_disableDigitalCompareCounterCapture
epwm.h	EPWM_setDigitalCompareCounterShadowMode
epwm.h	EPWM_getDigitalCompareCaptureStatus
<b>DCFOFFSET</b>	
epwm.h	EPWM_setDigitalCompareWindowOffset
epwm.h	EPWM_getDigitalCompareBlankingWindowOffsetCount
<b>DCFOFFSETCNT</b>	
epwm.h	EPWM_getDigitalCompareBlankingWindowOffsetCount
<b>DCFWINDOW</b>	
epwm.h	EPWM_setDigitalCompareWindowLength
epwm.h	EPWM_getDigitalCompareBlankingWindowLengthCount
<b>DCFWINDOWCNT</b>	
epwm.h	EPWM_getDigitalCompareBlankingWindowLengthCount
<b>DCCAP</b>	
epwm.h	EPWM_enableDigitalCompareCounterCapture
epwm.h	EPWM_disableDigitalCompareCounterCapture
epwm.h	EPWM_setDigitalCompareCounterShadowMode
epwm.h	EPWM_getDigitalCompareCaptureStatus
epwm.h	EPWM_getDigitalCompareCaptureCount
<b>DCAHTRIPSEL</b>	
epwm.h	EPWM_enableDigitalCompareTripCombinationInput
epwm.h	EPWM_disableDigitalCompareTripCombinationInput
<b>DCALTRIPSEL</b>	
-	See DCAHTRIPSEL
<b>DCBHTRIPSEL</b>	
-	See DCAHTRIPSEL
<b>DCBLTRIPSEL</b>	
-	See DCAHTRIPSEL
<b>LOCK</b>	
epwm.h	EPWM_lockRegisters
<b>HWVDELVAL</b>	
epwm.h	EPWM_getValleyHWDelay
<b>VCNTVAL</b>	
epwm.h	EPWM_getValleyCount

**Table 18-111. HRPWM Registers to Driverlib Functions**

File	Driverlib Function
<b>HRCNFG</b>	
hrpwm.h	HRPWM_setMEPEdgeSelect
hrpwm.h	HRPWM_setMEPControlMode
hrpwm.h	HRPWM_setCounterCompareShadowLoadEvent
hrpwm.h	HRPWM_setOutputSwapMode
hrpwm.h	HRPWM_setChannelBOutputPath
hrpwm.h	HRPWM_enableAutoConversion
hrpwm.h	HRPWM_disableAutoConversion
hrpwm.h	HRPWM_setDeadbandMEPEdgeSelect
hrpwm.h	HRPWM_setRisingEdgeDelayLoadMode
hrpwm.h	HRPWM_setFallingEdgeDelayLoadMode
<b>HRMSTEP</b>	
hrpwm.h	HRPWM_setMEPStep
<b>HRCNFG2</b>	
hrpwm.h	HRPWM_setDeadbandMEPEdgeSelect
hrpwm.h	HRPWM_setRisingEdgeDelayLoadMode
hrpwm.h	HRPWM_setFallingEdgeDelayLoadMode
<b>HRPCTL</b>	
hrpwm.h	HRPWM_enablePeriodControl
hrpwm.h	HRPWM_disablePeriodControl
hrpwm.h	HRPWM_enablePhaseShiftLoad
hrpwm.h	HRPWM_disablePhaseShiftLoad
hrpwm.h	HRPWM_setSyncPulseSource
<b>TRREM</b>	
hrpwm.h	HRPWM_setTranslatorRemainder
<b>DBREDHR</b>	
hrpwm.h	HRPWM_setRisingEdgeDelay
hrpwm.h	HRPWM_setHiResRisingEdgeDelayOnly
<b>DBRED</b>	
hrpwm.h	HRPWM_setRisingEdgeDelay
hrpwm.h	HRPWM_setHiResRisingEdgeDelayOnly
<b>DBFEDHR</b>	
hrpwm.h	HRPWM_setFallingEdgeDelay
hrpwm.h	HRPWM_setHiResFallingEdgeDelayOnly
<b>DBFED</b>	
hrpwm.h	HRPWM_setFallingEdgeDelay
hrpwm.h	HRPWM_setHiResFallingEdgeDelayOnly
<b>TBPHS</b>	
hrpwm.h	HRPWM_setPhaseShift
hrpwm.h	HRPWM_setHiResPhaseShiftOnly
<b>TBPRDHR</b>	
hrpwm.h	HRPWM_setTimeBasePeriod
hrpwm.h	HRPWM_setHiResTimeBasePeriodOnly
hrpwm.h	HRPWM_getTimeBasePeriod
hrpwm.h	HRPWM_getHiResTimeBasePeriodOnly
<b>TBPRD</b>	
hrpwm.h	HRPWM_setTimeBasePeriod
hrpwm.h	HRPWM_setHiResTimeBasePeriodOnly

**Table 18-111. HRPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
hrpwm.h	HRPWM_getTimeBasePeriod
hrpwm.h	HRPWM_getHiResTimeBasePeriodOnly
<b>CMPA</b>	
hrpwm.h	HRPWM_setCounterCompareValue
hrpwm.h	HRPWM_setHiResCounterCompareValueOnly
hrpwm.h	HRPWM_getCounterCompareValue
hrpwm.h	HRPWM_getHiResCounterCompareValueOnly
<b>CMPB</b>	
hrpwm.h	HRPWM_setCounterCompareValue
hrpwm.h	HRPWM_setHiResCounterCompareValueOnly
hrpwm.h	HRPWM_getCounterCompareValue
hrpwm.h	HRPWM_getHiResCounterCompareValueOnly
<b>EPWMLOCK</b>	
hrpwm.h	HRPWM_lockRegisters

## Enhanced Capture (eCAP)

---

---

This chapter describes the enhanced capture (eCAP) module, which is used in systems where accurate timing of external events is important.

Topic	Page
19.1 Introduction .....	2081
19.2 Features .....	2081
19.3 Description .....	2081
19.4 Configuring Device Pins for the eCAP .....	2082
19.5 Capture and APWM Operating Mode .....	2084
19.6 Capture Mode Description .....	2085
19.7 Application of the eCAP Module .....	2094
19.8 Application of the APWM Mode .....	2099
19.9 eCAP Registers .....	2100



## 19.1 Introduction

This eCAP module is a Type-1 eCAP. See the [TMS320C28xx, 28xxx DSP Peripheral Reference Guide](#) for a list of all devices with an eCAP module of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

## 19.2 Features

Features for eCAP include:

- Speed measurements of rotating machinery (for example, toothed sprockets sensed via Hall sensors)
- Elapsed time measurements between position sensor pulses
- Period and duty cycle measurements of pulse train signals
- Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors

The eCAP module described in this guide includes the following features:

- 4-event time-stamp registers (each 32 bits)
- Edge polarity selection for up to four sequenced time-stamp capture events
- Interrupt on either of the four events
- Single-shot capture of up to four event time-stamps
- Continuous mode capture of time stamps in a four-deep circular buffer
- Absolute time-stamp capture
- Difference (Delta) mode time-stamp capture
- When not used in capture mode, the eCAP module can be configured as a single-channel PWM output

The capture functionality of the Type-1 eCAP is enhanced from the Type-0 eCAP with the following added features:

- Event filter reset bit
  - Writing a 1 to ECCTL2[CTRFILTRESET] will clear the event filter, the modulo counter, and any pending interrupts flags. Resetting the bit is useful for initialization and debug.
- Modulo counter status bits
  - The modulo counter (ECCTL2 [MODCTRSTS]) indicates which capture register will be loaded next. In the Type-0 eCAP, it was not possible to know current state of modulo counter.
- DMA trigger source
  - eCAPxDMA was added as a DMA trigger. CEVT[1-4] can be configured as the source for eCAPxDMA.
- Input multiplexer
  - ECCTL0 [INPUTSEL] selects one of 128 input signals which are detailed in [Table 19-1](#).
- EALLOW protection
  - EALLOW protection was added to critical registers.

## 19.3 Description

The eCAP module represents one complete capture channel that can be instantiated multiple times, depending on the target device. In the context of this guide, one eCAP channel has the following independent key resources:

- Capture inputs can be connected using the Input X-BAR
- 128:1 input multiplexer
- Output X-BAR is used to configure output in APWM mode
- 32-bit time base (counter)
- 4 x 32-bit time-stamp capture registers (CAP1-CAP4)
- Four-stage sequencer (modulo4 counter) that is synchronized to external events, eCAP pin rising/falling edges.

- Modulo counter status register (MODCNTRSTS) to indicate sequencer state
- Independent edge polarity (rising/falling edge) selection for all four events
- Input capture signal prescaling (from 2-62 or bypass)
- One-shot compare register (two bits) to freeze captures after 1-4 time-stamp events
- Control for continuous time-stamp captures using a four-deep circular buffer (CAP1-CAP4) scheme
- Ability to reset event filter, modulo counter, and interrupt flags
- Interrupt capabilities on any of the four capture events
- Separate DMA trigger
- EALLOW protection to control registers

## 19.4 Configuring Device Pins for the eCAP

To connect the device input pins to the module, the Input X-BAR must be used. Any GPIO on the device can be configured as an input. The GPIO input qualification can be set to synchronous or asynchronous mode by setting the GPxQSELn register bits. Using synchronized inputs can help with noise immunity but will affect the eCAP's accuracy by  $\pm 2$  cycles. The internal pull-ups can be configured in the GPyPUD register. Since the GPIO mode is used, the GPyINV register can invert the signals.

New to the Type-1 eCAP module, a 128:1 input multiplexer must also be configured (see [Figure 19-3](#)). This multiplexer can select a variety of inputs detailed in [Table 19-1](#).

**Table 19-1. eCAP Input Selection**

Selection of eCAP Input	Select Index
Reserved	127
CMPSS[7:1].CTRIPL	126:120
Reserved	119:115
CMPSS[7:1].CTRIPL	114:108
Reserved	107:103
CMPSS[7:1].CTRIPL	102:96
Reserved	95:92
SD1FLT[4:1].COMPH_OR_COMPL	91:88
Reserved	87:84
SD1FLT[4:1].COMPH	83:80
Reserved	79:76
SD1FLT[4:1].COMPZ	75:72
Reserved	71:68
SD1FLT[4:1].COMPL	67:64
Reserved	63:51
Reserved	50
Reserved	49
Reserved	48
ADCAEVT[4:1]	47:44
ADCBEVT[4:1]	43:40
ADCCEVT[4:1]	39:36
Reserved	35:32
Output X-BAR OUTPUT[8:1]	31:24
Reserved	23:22
CANB_0 interrupt	21
CANA_0 interrupt	20
Reserved	19:16
Input X-BAR INPUT[16:1]	15:0

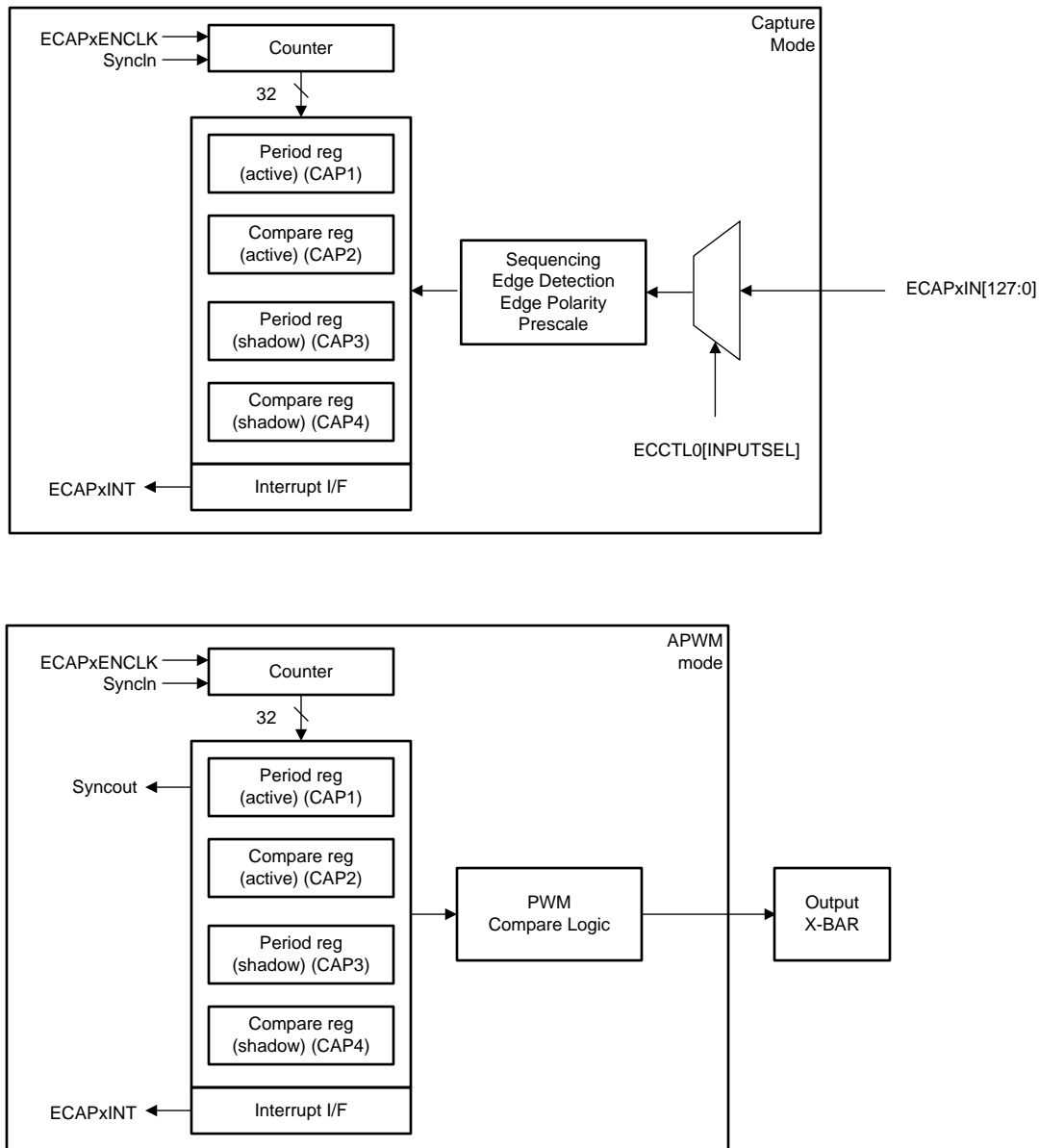
The Output X-BAR must be used to connect output signals to the OUTPUTXBARx output locations. The GPIO mux must then be configured to connect the OUTPUTXBARx lines to any of several IO pins with the GPIO mux. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

See the *GPIO Chapter* for more details on GPIO mux, GPIO settings, and XBAR configuration.

## 19.5 Capture and APWM Operating Mode

You can use the eCAP module resources to implement a single-channel PWM generator (with 32-bit capabilities) when it is not being used for input captures. The counter operates in count-up mode, providing a time-base for asymmetrical pulse width modulation (PWM) waveforms. The CAP1 and CAP2 registers become the active period and compare registers, respectively, while CAP3 and CAP4 registers become the period and capture shadow registers, respectively. Figure 19-1 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

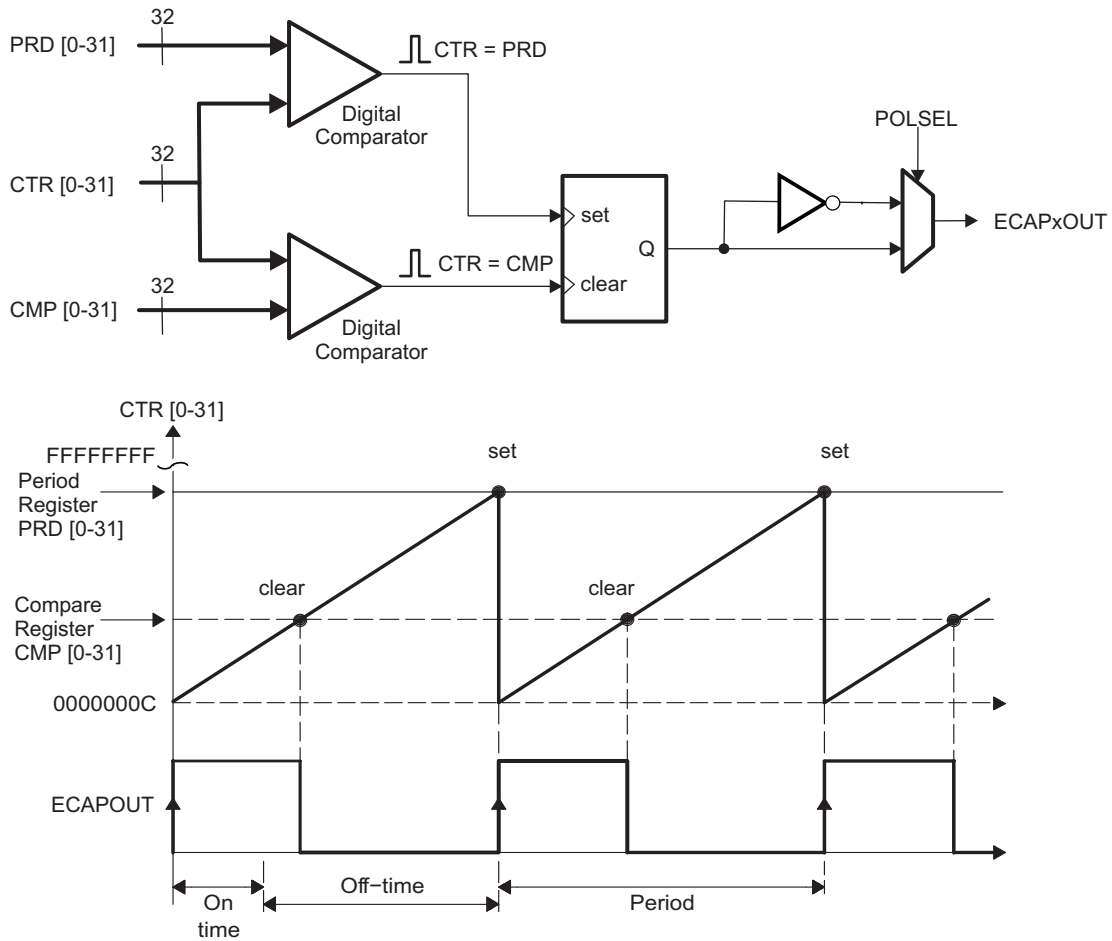
**Figure 19-1. Capture and APWM Modes of Operation**



- A A single pin is shared between CAP and APWM functions. In capture mode, it is an input; in APWM mode, it is an output.
- B In APWM mode, writing any value to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

Figure 19-2 further describes the output of the eCAP in APWM mode based on the CMP and PRD values.

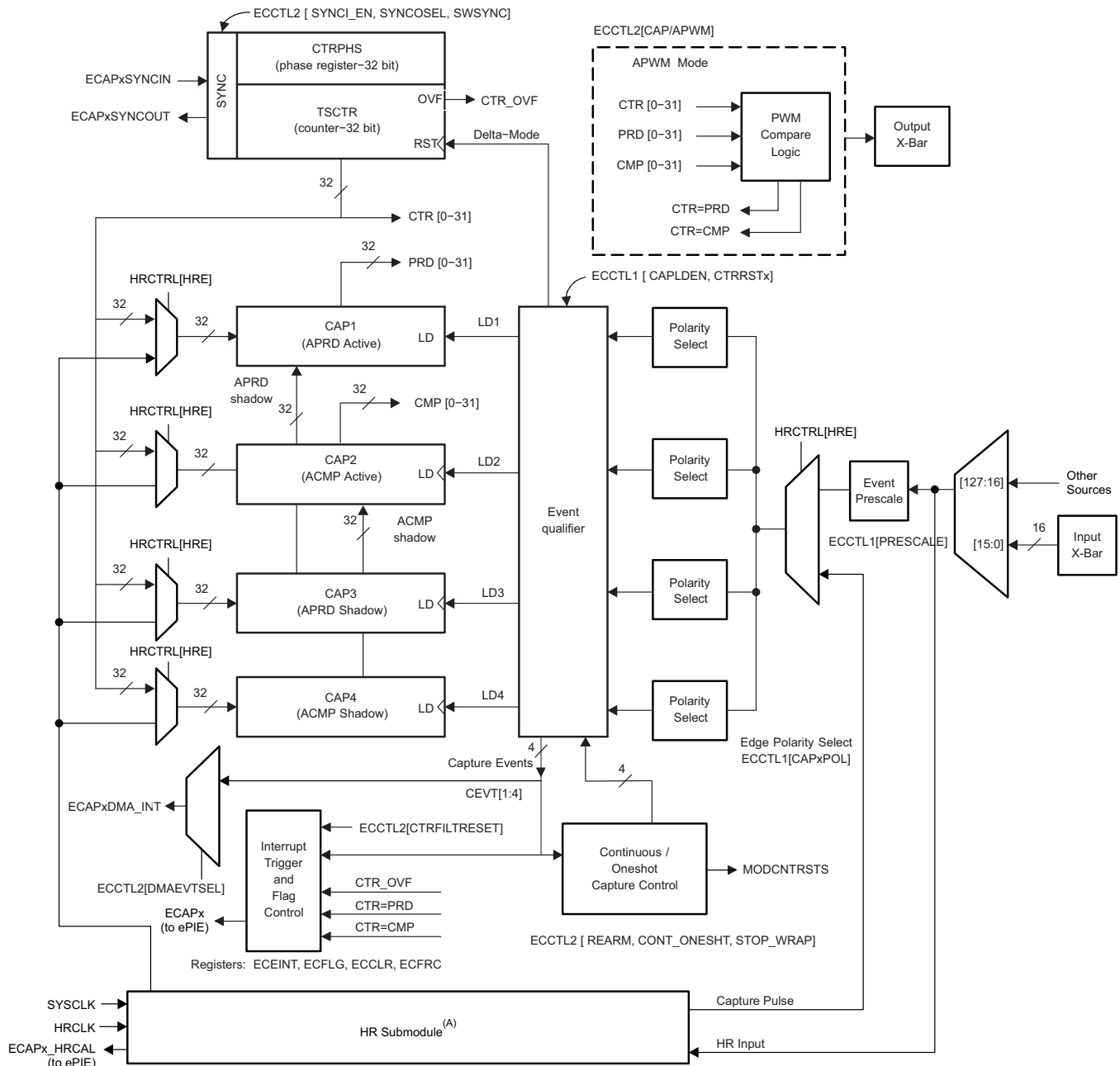
**Figure 19-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode**



## 19.6 Capture Mode Description

Figure 19-3 shows the various components that implement the capture function.

Figure 19-3. eCAP Block Diagram



Copyright © 2018, Texas Instruments Incorporated

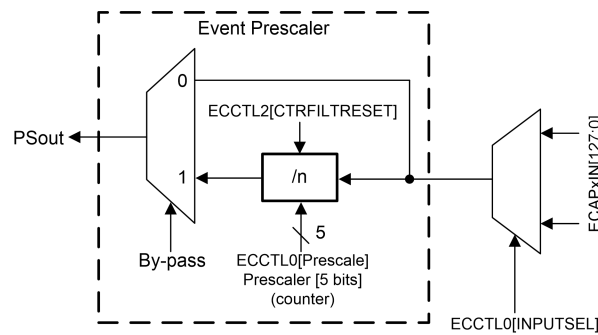
- A The HRCAP submodule is not available on all eCAP modules; in this case, the high-resolution muxes and hardware are not implemented.

### 19.6.1 Event Prescaler

- An input capture signal (pulse train) can be prescaled by  $N = 2-62$  (in multiples of 2) or can bypass the prescaler.

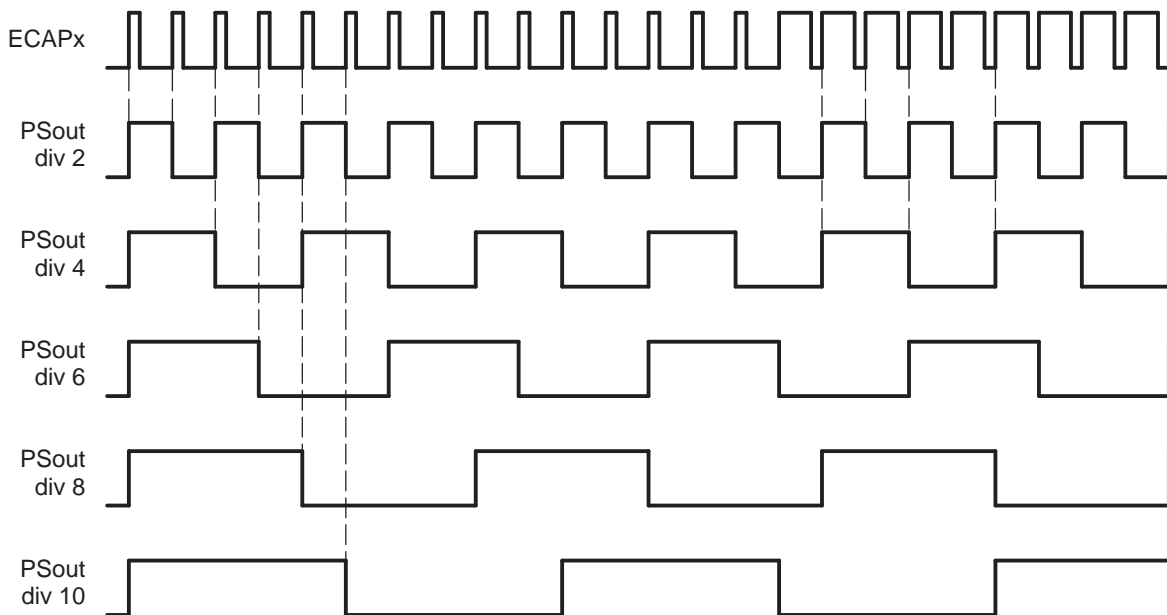
This is useful when very high frequency signals are used as inputs. Figure 19-4 shows a functional diagram and Figure 19-5 shows the operation of the prescale function. The event prescaler can be reset by setting the ECCTL2.CTRFILTRESET register bit.

Figure 19-4. Event Prescale Control



- A When a prescale value of 1 is chosen ( ECCTL1[13:9] = 0,0,0,0,0 ), the input capture signal bypasses the prescale logic completely.

Figure 19-5. Prescale Function Waveforms



### 19.6.2 Edge Polarity Select and Qualifier

Functionality and features include:

- Four independent edge polarity (rising edge/falling edge) selection muxes are used, one for each capture event.
- Each edge (up to 4) is event qualified by the Modulo4 sequencer.
- The edge event is gated to its respective CAPx register by the Mod4 counter. The CAPx register is loaded on the falling edge.

### 19.6.3 Continuous/One-Shot Control

Operation of eCAP in Continuous/One-Shot mode:

- The Mod4 (2-bit) counter is incremented via edge qualified events (CEVT1-CEVT4).
- The Mod4 counter continues counting (0->1->2->3->0) and wraps around unless stopped.
- A 2-bit stop register is used to compare the Mod4 counter output, and when equal, stops the Mod4 counter and inhibits further loads of the CAP1-CAP4 registers. This occurs during one-shot operation.

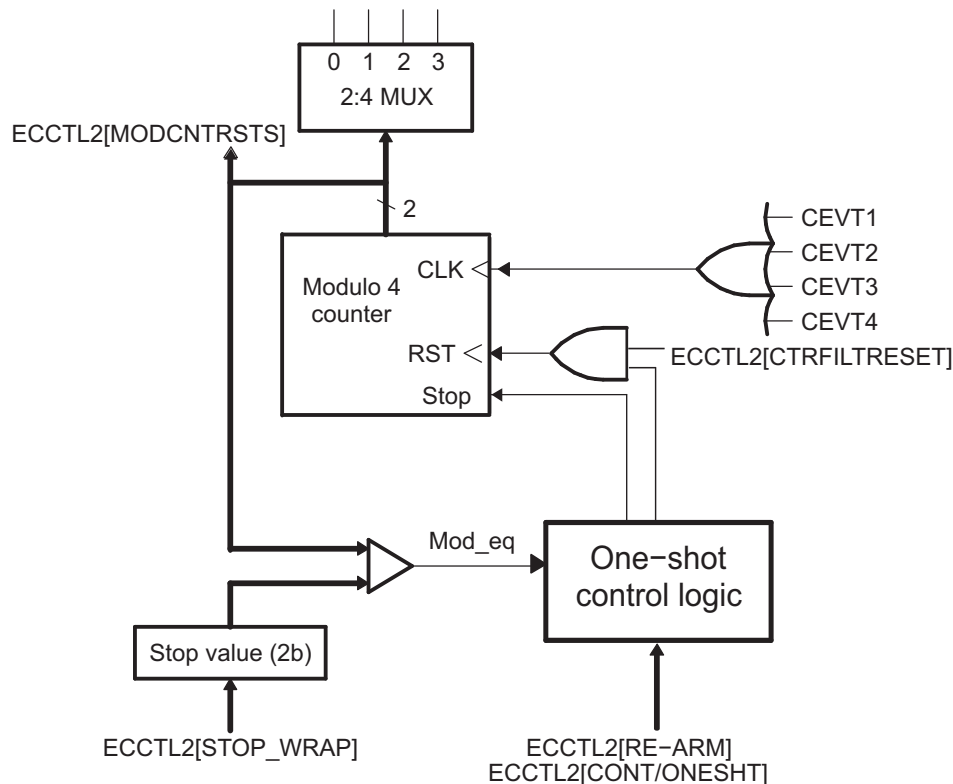
The continuous/one-shot block controls the start, stop and reset (zero) functions of the Mod4 counter, via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

Once armed, the eCAP module waits for 1-4 (defined by stop-value) capture events before freezing both the Mod4 counter and contents of CAP1-4 registers (time stamps).

Re-arming prepares the eCAP module for another capture sequence. Also, re-arming clears (to zero) the Mod4 counter and permits loading of CAP1-4 registers again, providing the CAPLDEN bit is set.

In continuous mode, the Mod4 counter continues to run (0->1->2->3->0, the one-shot action is ignored, and capture values continue to be written to CAP1-4 in a circular buffer sequence.

**Figure 19-6. Details of the Continuous/One-shot Block**



#### 19.6.4 32-Bit Counter and Phase Control

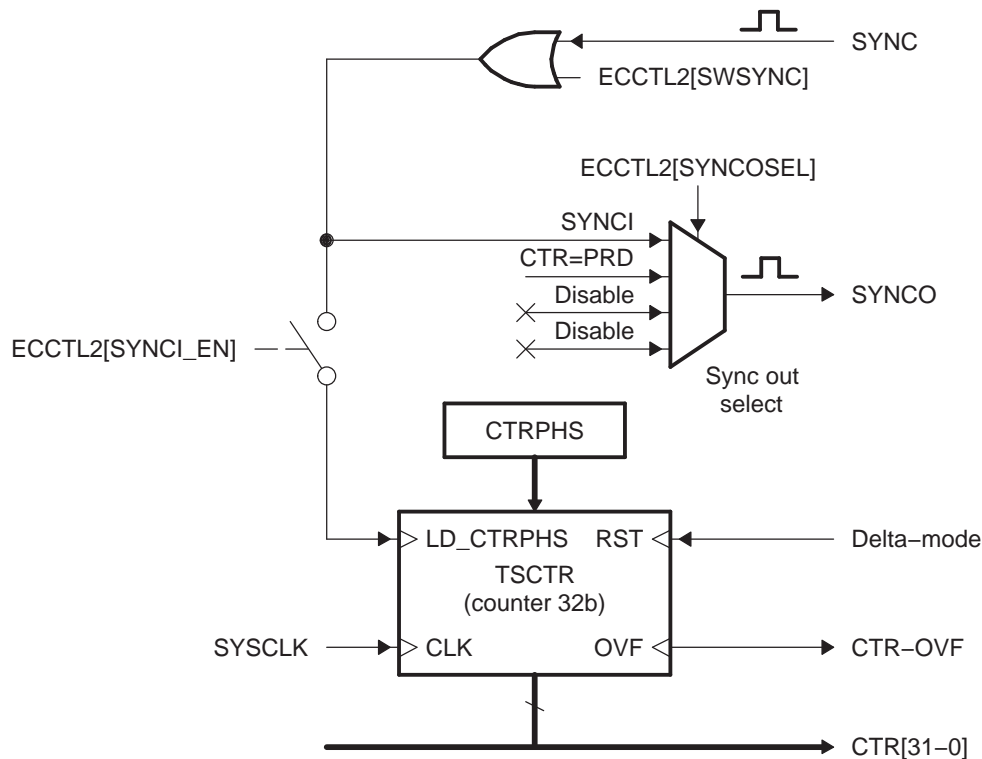
This counter provides the time-base for event captures, and is clocked via the system clock.

A phase register is provided to achieve synchronization with other counters, via a hardware and software forced sync. This is useful in APWM mode when a phase offset between modules is needed.

On any of the four event loads, an option to reset the 32-bit counter is given. This is useful for time difference capture. The 32-bit counter value is captured first, then it is reset to 0 by any of the LD1-LD4 signals.



Figure 19-7. Details of the Counter and Synchronization Block



### 19.6.5 CAP1-CAP4 Registers

These 32-bit registers are fed by the 32-bit counter timer bus, CTR[0-31] and are loaded (capture a timestamp) when their respective LD inputs are strobed.

Control bit CAPLDEN can inhibit loading of the capture registers. During one-shot operation, this bit is cleared (loading is inhibited) automatically when a stop condition occurs, StopValue = Mod4.

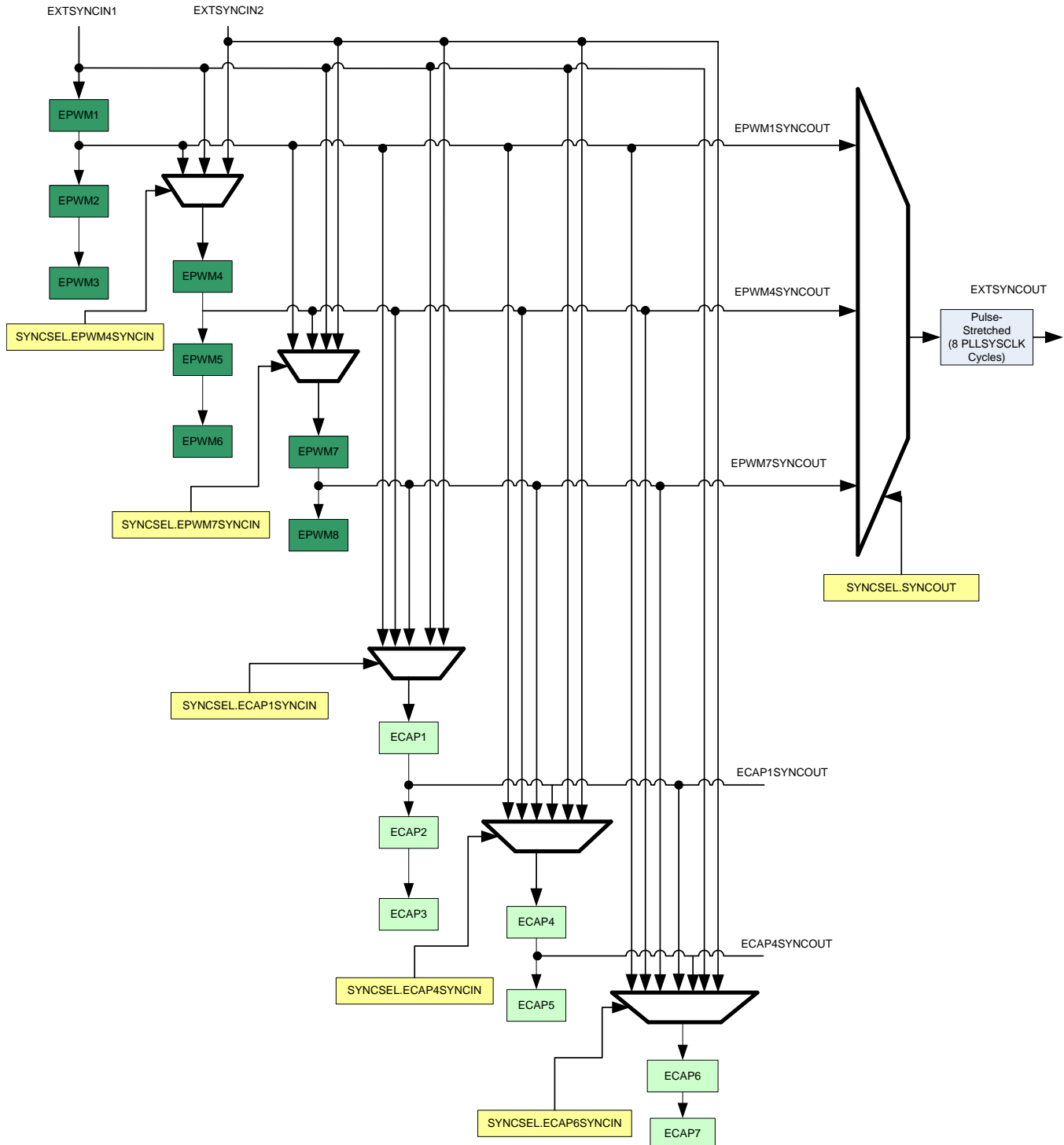
CAP1 and CAP2 registers become the active period and compare registers, respectively, in APWM mode.

CAP3 and CAP4 registers become the respective shadow registers (APRD and ACPM) for CAP1 and CAP2 during APWM operation.

### 19.6.6 eCAP Synchronization

eCAP modules can be synchronized with each other by selecting a common SYNCIN source. SYNCIN source for eCAP can be either software sync-in or external sync-in. The external sync-in signal can come from EPWM or eCAP or X-Bar. The SWSYNC of the eCAP module is logical OR'd with the SYNC signal as shown in Figure 19-7. The SYNC signal is defined by the selection in the SYNCSEL[ECAPxSYNCLN] bit for ECAP1 and ECAP4 as shown in Figure 19-8.

Figure 19-8. Time-Base Counter Synchronization Scheme



### 19.6.6.1 Example 1 - Using SWSYNC with eCAP Module

Implement the following steps to use SWSYNC with eCAP1 and eCAP3.

- Configure ECAP[1..3].ECAPSYNCINSEL.SEL = 0x0 to disable external SYNCIN coming to eCAP1.
- Configure ECAP[1..3].ECCTL2.SWSYNC = 0x1, to force Software Synchronization of the TSCTR counter.

- Configure ECAP[1..3].ECCTL2.SYNCO\_SEL = 0x0, to allow the sync-in event to be the sync-out signal pass through.
- Configure ECAP[2..3].ECCTL2.SWSYNC = 0x0, to disable software synchronization for eCAP2 through eCAP4.
- The default sync signal comes from ePWM1, if TBCTL[SYNCOSEL] is not correctly configured this can cause undesired resets of the time-stamp register (TSCTR). Select an unused GPIO in InputXbarRegs.INPUT5SELECT. Configure this GPIO in output mode and Write '0' to GPIO DAT register. By default this is programmed to GPIO0 so any activity on this pin will cause problems with the SWSYNC.
- Program SYNCSEL[ECAP1SYNCCIN] = 0x5. This will take ECAPx.EXTSYNCCIN to an inactive state.
- Configure ECAP1.ECCTL2.SWSYNC=0x1, this forces Software Synchronization of TSCTR counter

To use SWSYNC with other eCAP modules, ensure that the previous eCAP chain is not generating a SYNCOUT signal which will interfere with the software synchronization.

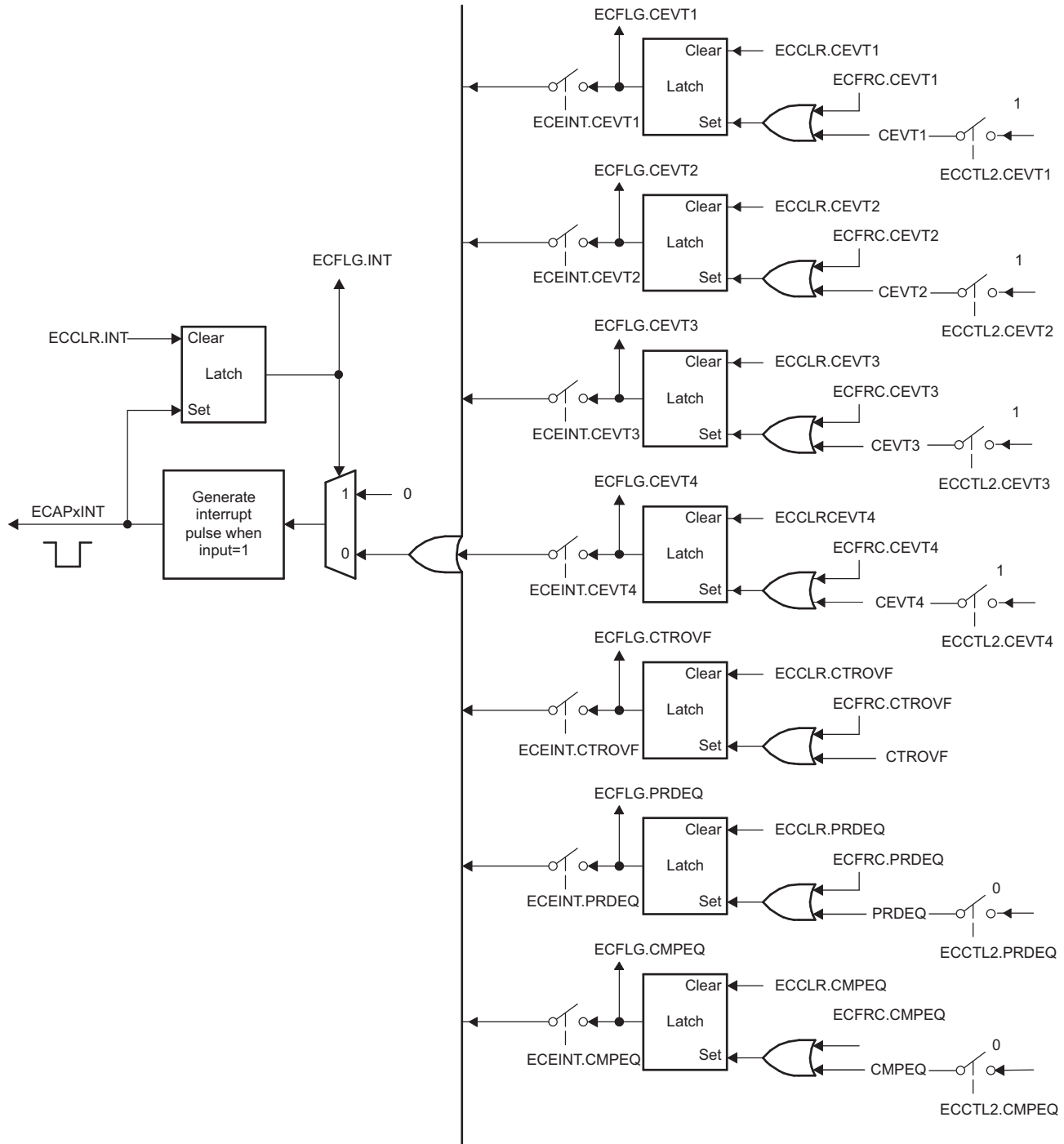
### 19.6.7 Interrupt Control

Operation and features of eCAP Interrupt Control include:

- An Interrupt can be generated on capture events (CEVT1-CEVT4, CTROVF) or APWM events (CTR = PRD, CTR = CMP).
- A counter overflow event (FFFFFFFF->00000000) is also provided as an interrupt source (CTROVF).
- The capture events are edge and sequencer-qualified (ordered in time) by the polarity select and Mod4 gating, respectively.
- One of these events can be selected as the interrupt source (from the eCAPx module) going to the PIE.
- Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, CTR=PRD, CTR=CMP) can be generated. The interrupt enable register (ECEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated to the PIE only if any of the interrupt events are enabled, the flag bit is 1, and the INT flag bit is 0. The interrupt service routine must clear the global interrupt flag bit and the serviced event via the interrupt clear register (ECCLR) before any other interrupt pulses are generated. All interrupt flags will be cleared upon an event filter reset by writing a "1" to ECCTL2[CLRFILTRESET]. You can force an interrupt event via the interrupt force register (ECFRC). This is useful for test purposes.

**Note:** The CEVT1, CEVT2, CEVT3, CEVT4 flags are only active in capture mode (ECCTL2[CAP/APWM == 0]). The CTR=PRD, CTR=CMP flags are only valid in APWM mode (ECCTL2[CAP/APWM == 1]). CNTOVF flag is valid in both modes.

Figure 19-9. Interrupts in eCAP Module



Copyright © 2018, Texas Instruments Incorporated

### 19.6.8 DMA Interrupt

On Type-0 eCAP modules, the CPU was required to begin data transfers using DMA. New to the Type-1 eCAP, a separate DMA Trigger (ECAP\_DMA\_INT) enables continuous transfer of capture data from eCAP registers to on-chip memory using DMA. Any one of the four available interrupt events (CEVT1, CEVT2, CEVT3 and CEVT4) can be selected as the trigger source for ECAP\_DMA\_INT using ECCCTL2 [DMAEVTSEL].

### 19.6.9 Shadow Load and Lockout Control

In capture mode, this logic inhibits (locks out) any shadow loading of CAP1 or CAP2 from APRD and ACMP registers, respectively.

In APWM mode, shadow loading is active and two choices are permitted:

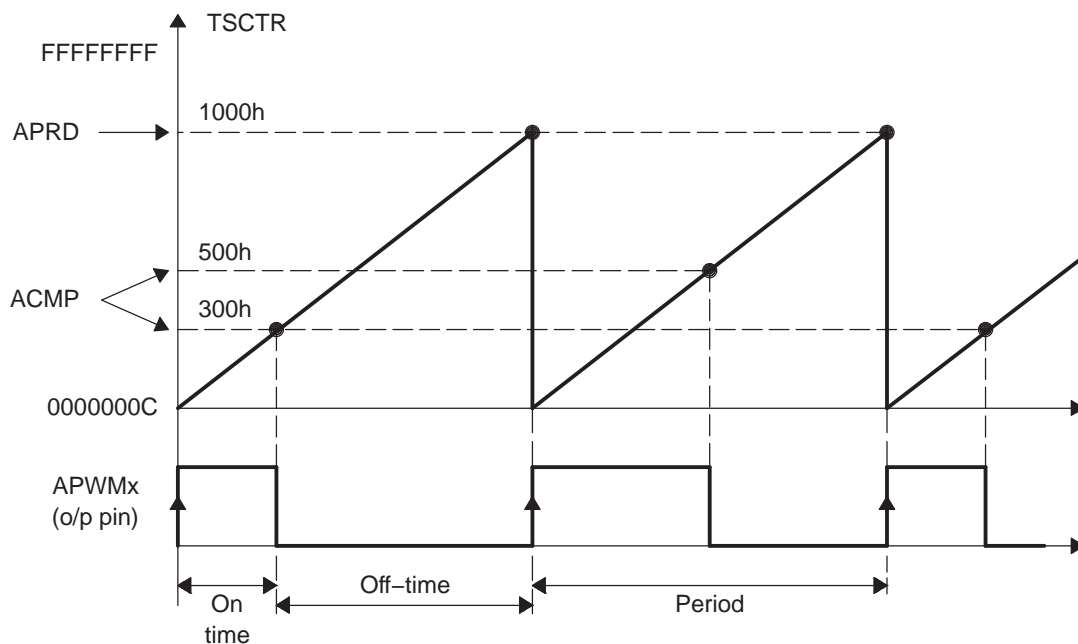
- Immediate - APRD or ACMP are transferred to CAP1 or CAP2 immediately upon writing a new value.
- On period equal,  $CTR[31:0] = PRD[31:0]$ .

### 19.6.10 APWM Mode Operation

Main operating highlights of the APWM section:

- The time-stamp counter bus is made available for comparison via 2 digital (32-bit) comparators.
- When CAP1/2 registers are not used in capture mode, their contents can be used as Period and Compare values in APWM mode.
- Double buffering is achieved via shadow registers APRD and ACMP (CAP3/4). The shadow register contents are transferred over to CAP1/2 registers, either immediately upon a write, or on a  $CTR = PRD$  trigger.
- In APWM mode, writing to CAP1/CAP2 active registers will also write the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 will invoke the shadow mode.
- During initialization, you must write to the active registers for both period and compare. This automatically copies the initial values into the shadow values. For subsequent compare updates, during run-time, you only need to use the shadow registers.

Figure 19-10. PWM Waveform Details Of APWM Mode Operation

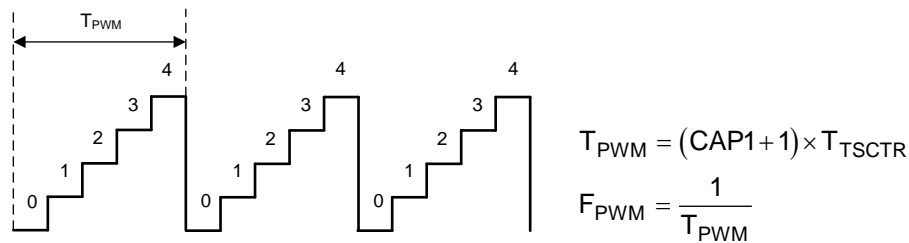


The behavior of APWM active high mode ( $APWMPOL == 0$ ) is as follows:

- CMP = 0x00000000, output low for duration of period (0% duty)
- CMP = 0x00000001, output high 1 cycle
- CMP = 0x00000002, output high 2 cycles
- CMP = PERIOD, output high except for 1 cycle (<100% duty)
- CMP = PERIOD+1, output high for complete period (100% duty)
- CMP > PERIOD+1, output high for complete period

The behavior of APWM active low mode ( $APWMPOL == 1$ ) is as follows:

CMP = 0x00000000, output high for duration of period (0% duty)  
 CMP = 0x00000001, output low 1 cycle  
 CMP = 0x00000002, output low 2 cycles  
 CMP = PERIOD, output low except for 1 cycle (<100% duty)  
 CMP = PERIOD+1, output low for complete period (100% duty)  
 CMP > PERIOD+1, output low for complete period

**Figure 19-11. Time-Base Frequency and Period Calculation**


## 19.7 Application of the eCAP Module

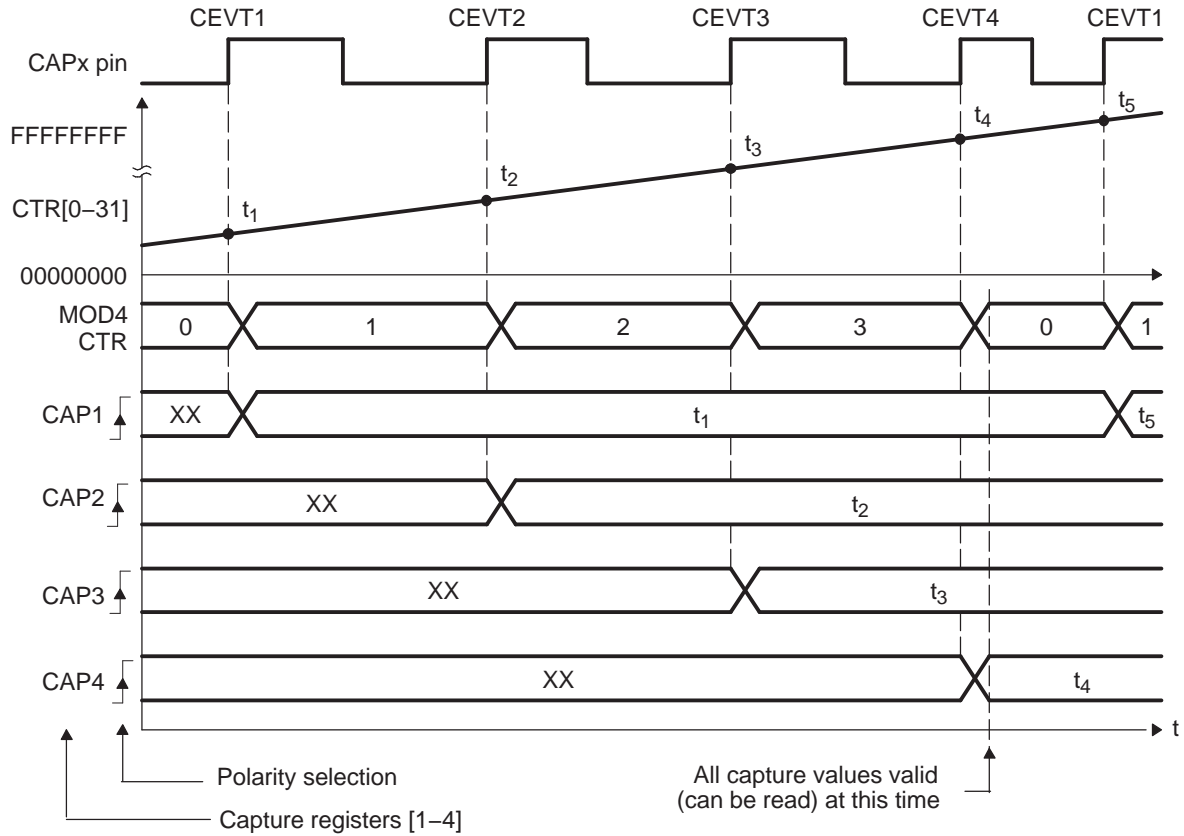
The following sections will provide applications examples to show how to operate the eCAP module.

### 19.7.1 Example 1 - Absolute Time-Stamp Operation Rising Edge Trigger

Figure 19-12 shows an example of continuous capture operation (Mod4 counter wraps around). In this figure, TSCTR counts-up without resetting and capture events are qualified on the rising edge only, this gives period (and frequency) information.

On an event, the TSCTR contents (time-stamp) is first captured, then Mod4 counter is incremented to the next state. When the TSCTR reaches FFFFFFFF (maximum value), it wraps around to 00000000 (not shown in Figure 19-12), if this occurs, the CTROVF (counter overflow) flag is set, and an interrupt (if enabled) occurs, CTROVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. Captured Time-stamps are valid at the point indicated by the diagram (after the 4th event), hence event CEVT4 can conveniently be used to trigger an interrupt and the CPU can read data from the CAPx registers.

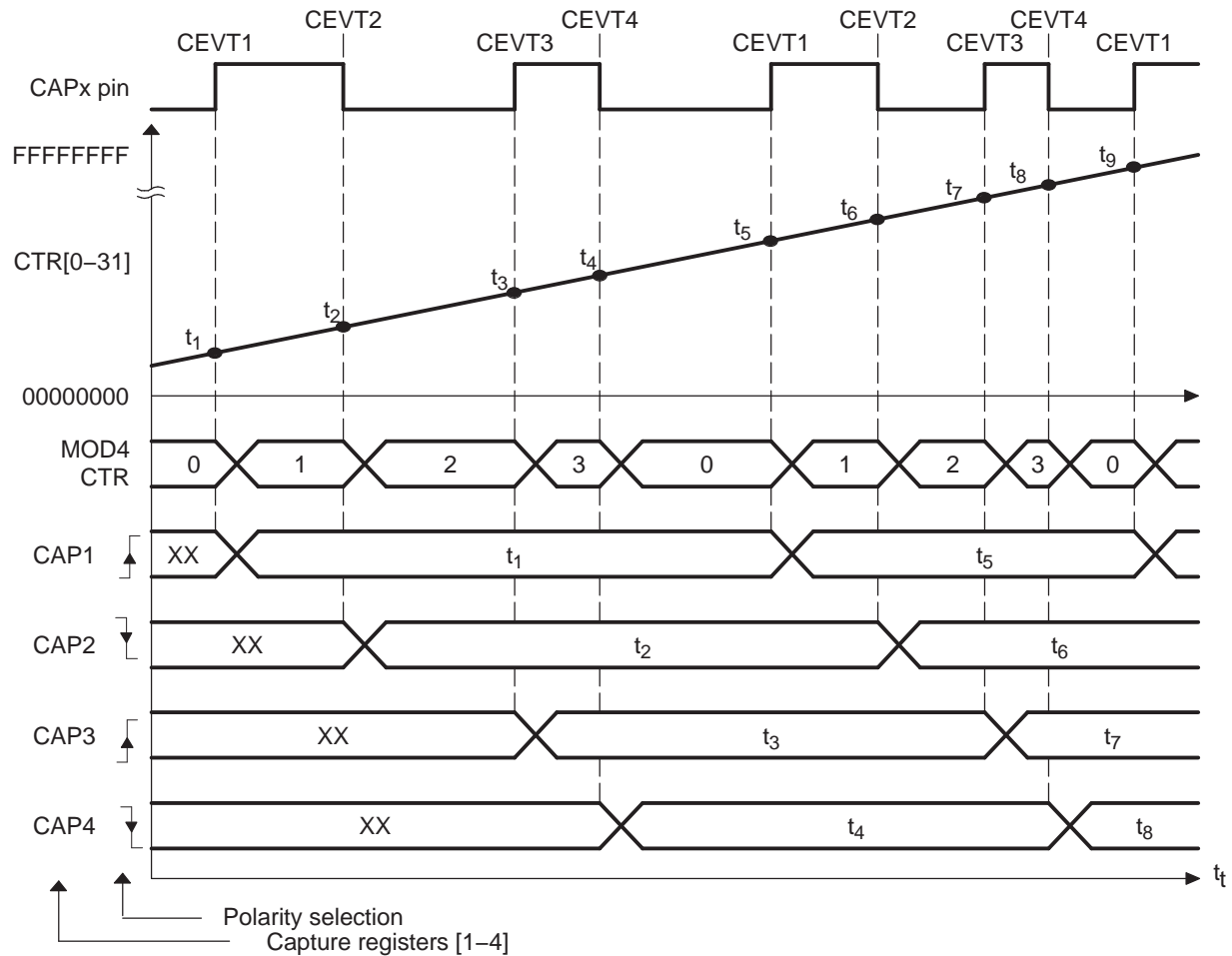
Figure 19-12. Capture Sequence for Absolute Time-stamp and Rising Edge Detect



### 19.7.2 Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger

In Figure 19-13 the eCAP operating mode is almost the same as in the previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information, that is:  $\text{Period1} = t_3 - t_1$ ,  $\text{Period2} = t_5 - t_3$ , ...and so on. Duty Cycle1 (on-time %) =  $(t_2 - t_1) / \text{Period1} \times 100\%$ , etc. Duty Cycle1 (off-time %) =  $(t_3 - t_2) / \text{Period1} \times 100\%$ , and so on.

**Figure 19-13. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect**

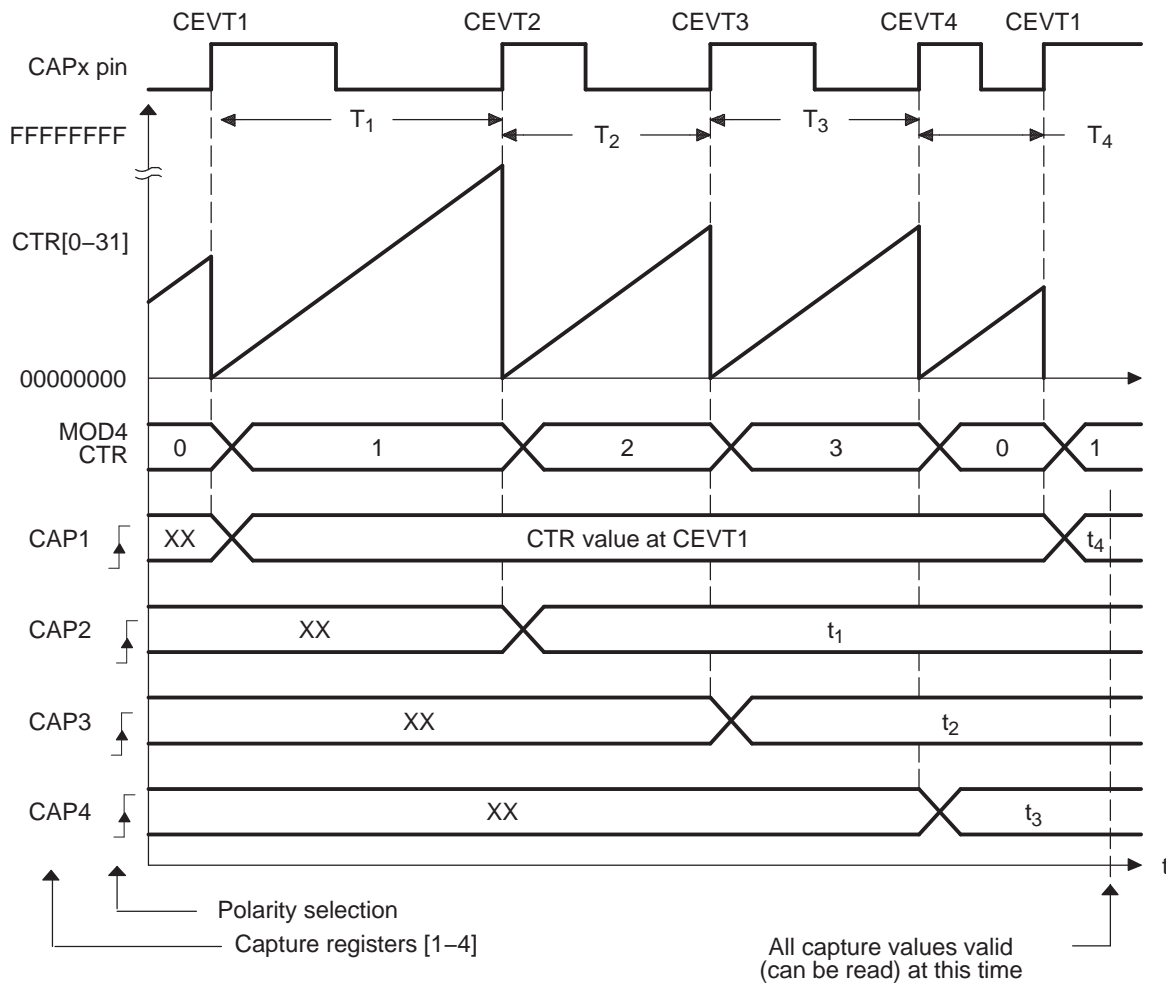




### 19.7.3 Example 3 - Time Difference (Delta) Operation Rising Edge Trigger

This example Figure 19-14 shows how the eCAP module can be used to collect Delta timing data from pulse train waveforms. Here Continuous Capture mode (TSCTR counts-up without resetting, and Mod4 counter wraps around) is used. In Delta-time mode, TSCTR is Reset back to Zero on every valid event. Here Capture events are qualified as Rising edge only. On an event, TSCTR contents (Time-Stamp) is captured first, and then TSCTR is reset to Zero. The Mod4 counter then increments to the next state. If TSCTR reaches FFFFFFFF (Max value), before the next event, it wraps around to 00000000 and continues, a CANTOVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. The advantage of Delta-time Mode is that the CAPx contents directly give timing data without the need for CPU calculations, that is,  $Period1 = T_1$ ,  $Period2 = T_2, \dots$  etc. As shown in the diagram, the CEVT1 event is a good trigger point to read the timing data,  $T_1, T_2, T_3, T_4$  are all valid here.

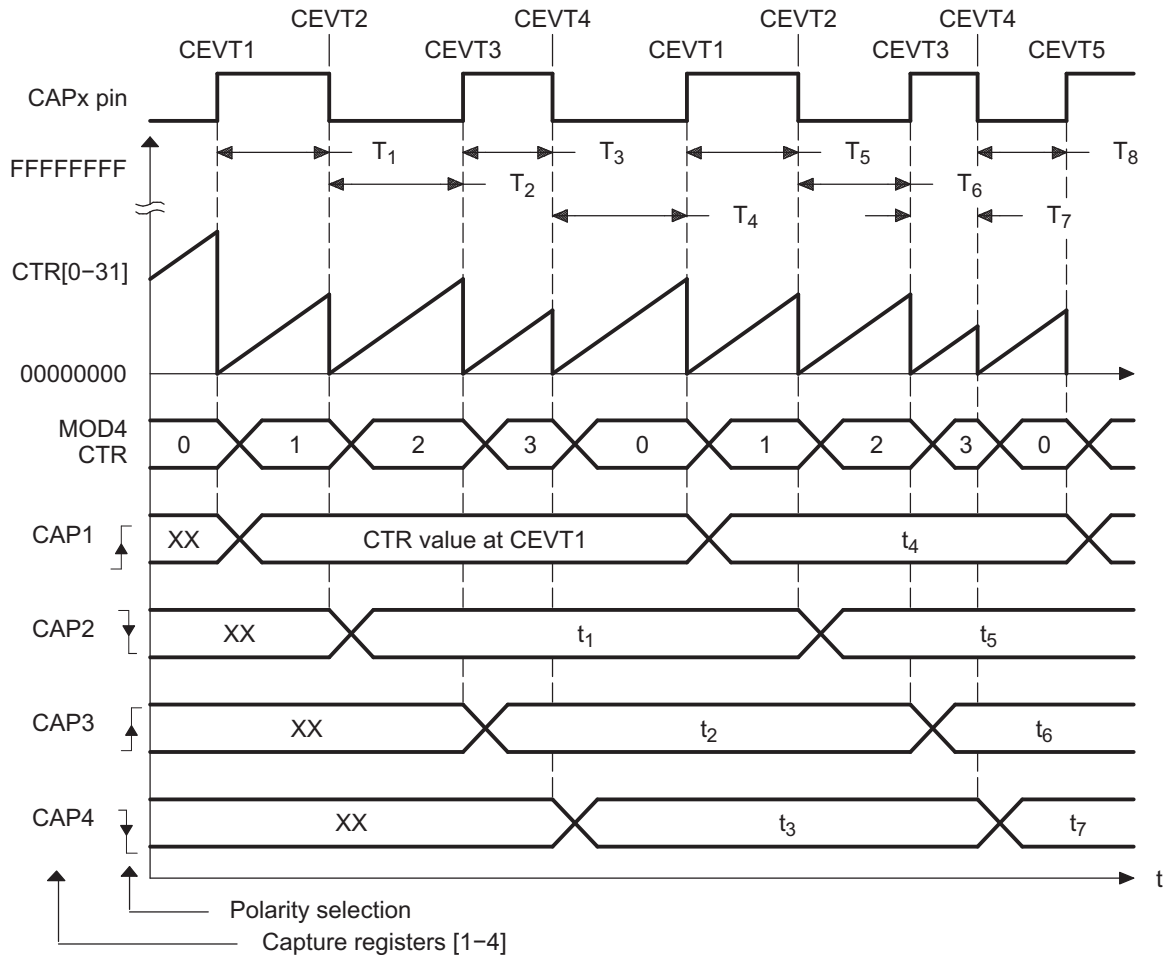
**Figure 19-14. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect**



### 19.7.4 Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger

In Figure 19-15 the eCAP operating mode is almost the same as in previous section except Capture events are qualified as either Rising or Falling edge, this now gives both Period and Duty cycle information, that is:  $Period1 = T_1 + T_2$ ,  $Period2 = T_3 + T_4$ , ...and so on,  $Duty Cycle1 (on-time \%) = T_1 / Period1 \times 100\%$ ,  $Duty Cycle1 (off-time \%) = T_2 / Period1 \times 100\%$ , and so on.

Figure 19-15. Capture Sequence for Delta Mode Time-stamp With Rising and Falling Edge Detect



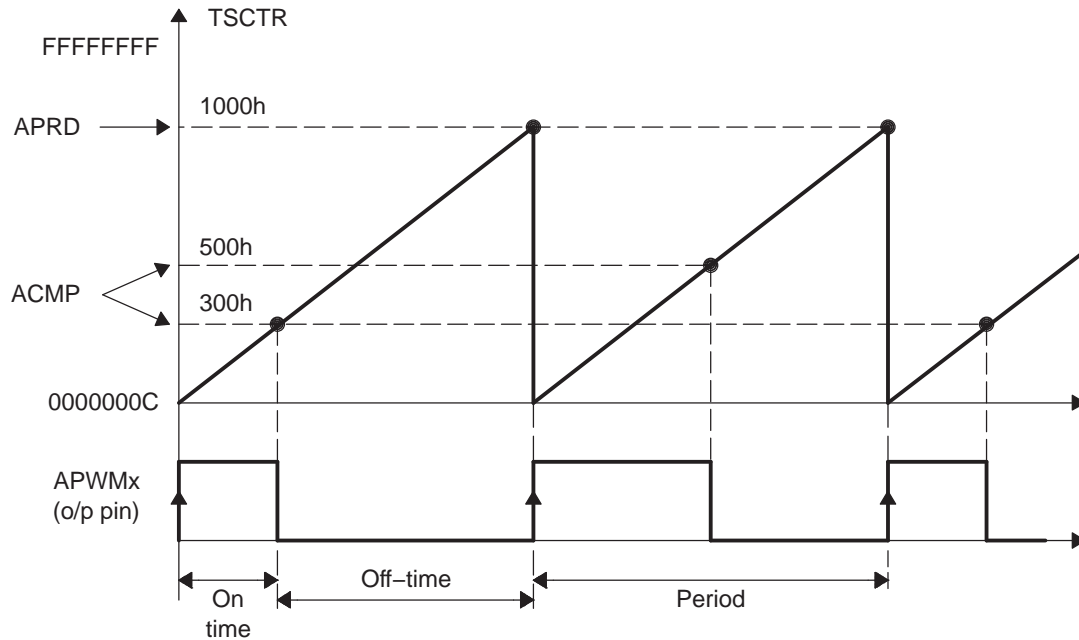
During initialization, you must write to the active registers for both period and compare. This action will automatically copy the init values into the shadow values. For subsequent compare updates during run-time, the shadow registers must be used.

## 19.8 Application of the APWM Mode

In this example, the eCAP module is configured to operate as a PWM generator. Here, a very simple single-channel PWM waveform is generated from the APWMx output pin. The PWM polarity is active high, which means that the compare value (CAP2 reg is now a compare register) represents the on-time (high level) of the period. Alternatively, if the APWMPOL bit is configured for active low, then the compare value represents the off-time.

### 19.8.1 Example 1 - Simple PWM Generation (Independent Channel/s)

Figure 19-16. PWM Waveform Details of APWM Mode Operation



**NOTE:** Values are in hexadecimal ("h") notation.

## 19.9 eCAP Registers

This section describes the Enhanced Capture Registers.

### 19.9.1 eCAP Base Addresses

**Table 19-2. eCAP Base Address Table**

Device Registers	Register Name	Start Address	End Address
ECap1Regs	ECAP_REGS	0x0000_5200	0x0000_521F
ECap2Regs	ECAP_REGS	0x0000_5240	0x0000_525F
ECap3Regs	ECAP_REGS	0x0000_5280	0x0000_529F
ECap4Regs	ECAP_REGS	0x0000_52C0	0x0000_52DF
ECap5Regs	ECAP_REGS	0x0000_5300	0x0000_531F
ECap6Regs	ECAP_REGS	0x0000_5340	0x0000_535F
ECap7Regs	ECAP_REGS	0x0000_5380	0x0000_539F

## 19.9.2 ECAP\_REGS Registers

Table 19-3 lists the ECAP\_REGS registers. All register offset addresses not listed in Table 19-3 should be considered as reserved locations and the register contents should not be modified.

**Table 19-3. ECAP\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	TSCTR	Time-Stamp Counter		<a href="#">Go</a>
2h	CTRPHS	Counter Phase Offset Value Register		<a href="#">Go</a>
4h	CAP1	Capture 1 Register		<a href="#">Go</a>
6h	CAP2	Capture 2 Register		<a href="#">Go</a>
8h	CAP3	Capture 3 Register		<a href="#">Go</a>
Ah	CAP4	Capture 4 Register		<a href="#">Go</a>
12h	ECCTL0	Capture Control Register 0	EALLOW	<a href="#">Go</a>
14h	ECCTL1	Capture Control Register 1	EALLOW	<a href="#">Go</a>
15h	ECCTL2	Capture Control Register 2	EALLOW	<a href="#">Go</a>
16h	ECEINT	Capture Interrupt Enable Register	EALLOW	<a href="#">Go</a>
17h	ECFLG	Capture Interrupt Flag Register		<a href="#">Go</a>
18h	ECCLR	Capture Interrupt Clear Register		<a href="#">Go</a>
19h	ECFRC	Capture Interrupt Force Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 19-4 shows the codes that are used for access types in this section.

**Table 19-4. ECAP\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1C	W1C	Write 1 to clear
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 19.9.2.1 TSCTR Register (Offset = 0h) [reset = 0h]

TSCTR is shown in [Figure 19-17](#) and described in [Table 19-5](#).

Return to the [Summary Table](#).

Time-Stamp Counter

**Figure 19-17. TSCTR 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
TSCTR																															
R/W-0h																															

**Table 19-5. TSCTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	TSCTR	R/W	0h	Active 32-bit counter register that is used as the capture time-base HR mode : 1) This register reads HRCOUNTER value and is not writable 2) can be reset using CTRFILTRESET 3) Its not synchronized to SYSCLK domain so reads may not be accurate Reset type: SYSRSn

### 19.9.2.2 CTRPHS Register (Offset = 2h) [reset = 0h]

CTRPHS is shown in [Figure 19-18](#) and described in [Table 19-6](#).

Return to the [Summary Table](#).

Counter Phase Offset Value Register

**Figure 19-18. CTRPHS 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
CTRPHS																															
R/W-0h																															

**Table 19-6. CTRPHS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CTRPHS	R/W	0h	<p>Counter phase value register that can be programmed for phase lag/lead. This register CTRPHS is loaded into TSCTR upon either a SYNCI event or S/W force via a control bit. Used to achieve phase control synchronization with respect to other eCAP and EPWM time-bases.</p> <p>This register is not applicable in HR mode.</p> <p>Reset type: SYSRSn</p>

### 19.9.2.3 CAP1 Register (Offset = 4h) [reset = 0h]

CAP1 is shown in [Figure 19-19](#) and described in [Table 19-7](#).

Return to the [Summary Table](#).

Capture 1 Register

**Figure 19-19. CAP1 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
CAP1																															
R/W-0h																															

**Table 19-7. CAP1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CAP1	R/W	0h	This register can be loaded (written) by: <ul style="list-style-type: none"> <li>- Time-Stamp counter value (TSCTR) during a capture event</li> <li>- Software - may be useful for test purposes or initialization</li> <li>- ARPD shadow register (CAP3) when used in APWM mode</li> </ul> Reset type: SYSRSn



#### 19.9.2.4 CAP2 Register (Offset = 6h) [reset = 0h]

CAP2 is shown in [Figure 19-20](#) and described in [Table 19-8](#).

Return to the [Summary Table](#).

Capture 2 Register

**Figure 19-20. CAP2 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
CAP2																															
R/W-0h																															

**Table 19-8. CAP2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CAP2	R/W	0h	This register can be loaded (written) by: <ul style="list-style-type: none"> <li>- Time-Stamp ( counter value) during a capture event</li> <li>- Software - may be useful for test purposes</li> <li>- ACMP shadow register (CAP4) when used in APWM mode</li> </ul> Reset type: SYSRSn

### 19.9.2.5 CAP3 Register (Offset = 8h) [reset = 0h]

CAP3 is shown in [Figure 19-21](#) and described in [Table 19-9](#).

Return to the [Summary Table](#).

Capture 3 Register

**Figure 19-21. CAP3 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
CAP3																															
R/W-0h																															

**Table 19-9. CAP3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CAP3	R/W	0h	In CMP mode, this is a time-stamp capture register.  In APWM mode, this is the period shadow (APRD) register. You can update the PWM period value through this register. CAP3 (APRD) shadows CAP1 in this mode.  Reset type: SYSRSn

### 19.9.2.6 CAP4 Register (Offset = Ah) [reset = 0h]

CAP4 is shown in [Figure 19-22](#) and described in [Table 19-10](#).

Return to the [Summary Table](#).

Capture 4 Register

**Figure 19-22. CAP4 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
CAP4																															
R/W-0h																															

**Table 19-10. CAP4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CAP4	R/W	0h	<p>In CMP mode, this is a time-stamp capture register.</p> <p>In APWM mode, this is the compare shadow (ACMP) register. You can update the PWM compare value via this register. CAP4 (ACMP) shadows CAP2 in this mode.</p> <p>Reset type: SYSRSn</p>

**19.9.2.7 ECCTL0 Register (Offset = 12h) [reset = 7Fh]**

ECCTL0 is shown in [Figure 19-23](#) and described in [Table 19-11](#).

Return to the [Summary Table](#).

Capture Control Register 0

**Figure 19-23. ECCTL0 Register**

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

**Table 19-11. ECCTL0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-7	RESERVED	R-0	0h	Reserved
6-0	INPUTSEL	R/W	7Fh	Capture input source select bits 0000000 capture input is ECAPxINPUT[0] 0000001 capture input is ECAPxINPUT[1] 0000010 capture input is ECAPxINPUT[2] ... 1111111 capture input is ECAPxINPUT[127] Reset type: CPU1.SYSRSn

### 19.9.2.8 ECCTL1 Register (Offset = 14h) [reset = 0h]

ECCTL1 is shown in [Figure 19-24](#) and described in [Table 19-12](#).

Return to the [Summary Table](#).

Capture Control Register 1

**Figure 19-24. ECCTL1 Register**

15		14		13		12		11		10		9		8	
FREE_SOFT				PRESCALE								CAPLDEN			
R/W-0h				R/W-0h								R/W-0h			
7		6		5		4		3		2		1		0	
CTRRST4		CAP4POL		CTRRST3		CAP3POL		CTRRST2		CAP2POL		CTRRST1		CAP1POL	
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 19-12. ECCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation Control Reset type: SYSRSn 0h (R/W) = TSCTR counter stops immediately on emulation suspend 1h (R/W) = TSCTR counter runs until = 0 2h (R/W) = TSCTR counter is unaffected by emulation suspend (Run Free) 3h (R/W) = TSCTR counter is unaffected by emulation suspend (Run Free)
13-9	PRESCALE	R/W	0h	Event Filter prescale select Reset type: SYSRSn 0h (R/W) = Divide by 1 (i.e., no prescale, by-pass the prescaler) 1h (R/W) = Divide by 2 2h (R/W) = Divide by 4 3h (R/W) = Divide by 6 4h (R/W) = Divide by 8 5h (R/W) = Divide by 10 1Eh (R/W) = Divide by 60 1Fh (R/W) = Divide by 62
8	CAPLDEN	R/W	0h	Enable Loading of CAP1-4 registers on a capture event. Note that this bit does not disable CEVTn events from being generated. Reset type: SYSRSn 0h (R/W) = Disable CAP1-4 register loads at capture event time. 1h (R/W) = Enable CAP1-4 register loads at capture event time.
7	CTRRST4	R/W	0h	Counter Reset on Capture Event 4 Reset type: SYSRSn 0h (R/W) = Do not reset counter on Capture Event 4 (absolute time stamp operation) 1h (R/W) = Reset counter after Capture Event 4 time-stamp has been captured (used in difference mode operation)
6	CAP4POL	R/W	0h	Capture Event 4 Polarity select Reset type: SYSRSn 0h (R/W) = Capture Event 4 triggered on a rising edge (RE) 1h (R/W) = Capture Event 4 triggered on a falling edge (FE)

**Table 19-12. ECCTL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	CTRRST3	R/W	0h	Counter Reset on Capture Event 3 Reset type: SYSRSn 0h (R/W) = Do not reset counter on Capture Event 3 (absolute time stamp) 1h (R/W) = Reset counter after Event 3 time-stamp has been captured (used in difference mode operation)
4	CAP3POL	R/W	0h	Capture Event 3 Polarity select Reset type: SYSRSn 0h (R/W) = Capture Event 3 triggered on a rising edge (RE) 1h (R/W) = Capture Event 3 triggered on a falling edge (FE)
3	CTRRST2	R/W	0h	Counter Reset on Capture Event 2 Reset type: SYSRSn 0h (R/W) = Do not reset counter on Capture Event 2 (absolute time stamp) 1h (R/W) = Reset counter after Event 2 time-stamp has been captured (used in difference mode operation)
2	CAP2POL	R/W	0h	Capture Event 2 Polarity select Reset type: SYSRSn 0h (R/W) = Capture Event 2 triggered on a rising edge (RE) 1h (R/W) = Capture Event 2 triggered on a falling edge (FE)
1	CTRRST1	R/W	0h	Counter Reset on Capture Event 1 Reset type: SYSRSn 0h (R/W) = Do not reset counter on Capture Event 1 (absolute time stamp) 1h (R/W) = Reset counter after Event 1 time-stamp has been captured (used in difference mode operation)
0	CAP1POL	R/W	0h	Capture Event 1 Polarity select Reset type: SYSRSn 0h (R/W) = Capture Event 1 triggered on a rising edge (RE) 1h (R/W) = Capture Event 1 triggered on a falling edge (FE)

### 19.9.2.9 ECCTL2 Register (Offset = 15h) [reset = 6h]

ECCTL2 is shown in [Figure 19-25](#) and described in [Table 19-13](#).

Return to the [Summary Table](#).

Capture Control Register 2

**Figure 19-25. ECCTL2 Register**

15	14	13	12	11	10	9	8
MODCNRSTS		DMAEVTSEL		CTRFILTRESE T	APWMPOL	CAP_APWM	SWSYNC
R-0h		R/W-0h		R-0/W1C-0h	R/W-0h	R/W-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
SYNCO_SEL		SYNCL_EN	TSCTRSTOP	REARM	STOP_WRAP		CONT_ONESH T
R/W-0h		R/W-0h	R/W-0h	R-0/W1S-0h	R/W-3h		R/W-0h

**Table 19-13. ECCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	MODCNRSTS	R	0h	This bit field reads current status on modulo counter 00b (R) = CAP1 register gets loaded on next capture event. 01b (R) = CAP2 register gets loaded on next capture event. 10b (R) = CAP3 register gets loaded on next capture event. 11b (R) = CAP4 register gets loaded on next capture event. Reset type: CPU1.SYSRSn
13-12	DMAEVTSEL	R/W	0h	DMA event select 00b (R/W) = DMA interrupt source is CEVT1 01b (R/W) = DMA interrupt source is CEVT2 10b (R/W) = DMA interrupt source is CEVT3 11b (R/W) = DMA interrupt source is CEVT4 Reset type: CPU1.SYSRSn
11	CTRFILTRESET	R-0/W1C	0h	Reset Bit 0h (R) = No effect 1h (W) = Resets event filter, counter, modulo counter and CEVT[1,2,3,4] and CNTOVF , HRERROR flags  Note: This provides an ability start capture module from known state in case spurious inputs are captured while ECAP is configured. Reset type: CPU1.SYSRSn
10	APWMPOL	R/W	0h	APWM output polarity select. This is applicable only in APWM operating mode. Reset type: SYSRSn 0h (R/W) = Output is active high (Compare value defines high time) 1h (R/W) = Output is active low (Compare value defines low time)
9	CAP_APWM	R/W	0h	CAP/APWM operating mode select Reset type: SYSRSn 0h (R/W) = ECAP module operates in capture mode. This mode forces the following configuration: - Inhibits TSCTR resets via CTR = PRD event - Inhibits shadow loads on CAP1 and 2 registers - Permits user to enable CAP1-4 register load - CAPx/APWMx pin operates as a capture input 1h (R/W) = ECAP module operates in APWM mode. This mode forces the following configuration: - Resets TSCTR on CTR = PRD event (period boundary) - Permits shadow loading on CAP1 and 2 registers - Disables loading of time-stamps into CAP1-4 registers - CAPx/APWMx pin operates as a APWM output

**Table 19-13. ECCTL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	SWSYNC	R-0/W1S	0h	Software-forced Counter (TSCTR) Synchronizer. This provides the user a method to generate a synchronization pulse through software. In APWM mode, the synchronization pulse can also be sourced from the CTR = PRD event. Reset type: SYSRSn 0h (R/W) = Writing a zero has no effect. Reading always returns a zero 1h (R/W) = Writing a one forces a TSCTR shadow load of current ECAP module and any ECAP modules down-stream providing the SYNCO_SEL bits are 0,0. After writing a 1, this bit returns to a zero. Note: Selection CTR = PRD is meaningful only in APWM mode however, you can choose it in CAP mode if you find doing so useful.
7-6	SYNCO_SEL	R/W	0h	Sync-Out Select Reset type: SYSRSn 0h (R/W) = Select sync-in event to be the sync-out signal pass through 1h (R/W) = Select CTR = PRD event to be the sync-out signal 2h (R/W) = Disable sync out signal 3h (R/W) = Disable sync out signal
5	SYNCl_EN	R/W	0h	Counter (TSCTR) Sync-In select mode Reset type: SYSRSn 0h (R/W) = Disable sync-in option 1h (R/W) = Enable counter (TSCTR) to be loaded from CTRPHS register upon either a SYNCl signal or a S/W force event.
4	TSCTRSTOP	R/W	0h	Time Stamp (TSCTR) Counter Stop (freeze) Control Reset type: SYSRSn 0h (R/W) = TSCTR stopped 1h (R/W) = TSCTR free-running
3	REARM	R-0/W1S	0h	Re-Arming Control. Note: The re-arm function is valid in one shot or continuous mode Reset type: SYSRSn 0h (R/W) = Has no effect (reading always returns a 0) 1h (R/W) = Arms the one-shot sequence as follows: 1) Resets the Mod4 counter to zero 2) Unfreezes the Mod4 counter 3) Enables capture register loads



**Table 19-13. ECCTL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2-1	STOP_WRAP	R/W	3h	<p>Stop value for one-shot mode. This is the number (between 1-4) of captures allowed to occur before the CAP(1-4) registers are frozen, that is, capture sequence is stopped.</p> <p>Wrap value for continuous mode. This is the number (between 1-4) of the capture register in which the circular buffer wraps around and starts again.</p> <p>Notes: STOP_WRAP is compared to Mod4 counter and, when equal, 2 actions occur:</p> <ul style="list-style-type: none"> <li>- Mod4 counter is stopped (frozen)</li> <li>- Capture register loads are inhibited</li> </ul> <p>In one-shot mode, further interrupt events are blocked until re-armed.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Stop after Capture Event 1 in one-shot mode Wrap after Capture Event 1 in continuous mode.</p> <p>1h (R/W) = Stop after Capture Event 2 in one-shot mode Wrap after Capture Event 2 in continuous mode.</p> <p>2h (R/W) = Stop after Capture Event 3 in one-shot mode Wrap after Capture Event 3 in continuous mode.</p> <p>3h (R/W) = Stop after Capture Event 4 in one-shot mode Wrap after Capture Event 4 in continuous mode.</p>
0	CONT_ONESHT	R/W	0h	<p>Continuous or one-shot mode control (applicable only in capture mode)</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Operate in continuous mode</p> <p>1h (R/W) = Operate in one-Shot mode</p>

### 19.9.2.10 ECEINT Register (Offset = 16h) [reset = 0h]

ECEINT is shown in [Figure 19-26](#) and described in [Table 19-14](#).

Return to the [Summary Table](#).

The interrupt enable bits (CEVT1, ...) block any of the selected events from generating an interrupt. Events will still be latched into the flag bit (ECFLG register) and can be forced/cleared via the ECFRC/ECCLR registers.

The proper procedure for configuring peripheral modes and interrupts is as follows:

- Disable global interrupts
- Stop eCAP counter
- Disable eCAP interrupts
- Configure peripheral registers
- Clear spurious eCAP interrupt flags
- Enable eCAP interrupts
- Start eCAP counter
- Enable global interrupts

**Figure 19-26. ECEINT Register**

15	14	13	12	11	10	9	8
RESERVED							RESERVED
R-0h							
7	6	5	4	3	2	1	0
CTR_EQ_CMP	CTR_EQ_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	RESERVED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

**Table 19-14. ECEINT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	CTR_EQ_CMP	R/W	0h	Counter Equal Compare Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Compare Equal as an Interrupt source 1h (R/W) = Enable Compare Equal as an Interrupt source
6	CTR_EQ_PRD	R/W	0h	Counter Equal Period Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Period Equal as an Interrupt source 1h (R/W) = Enable Period Equal as an Interrupt source
5	CTROVF	R/W	0h	Counter Overflow Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disabled counter Overflow as an Interrupt source 1h (R/W) = Enable counter Overflow as an Interrupt source
4	CEVT4	R/W	0h	Capture Event 4 Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Capture Event 4 as an Interrupt source 1h (R/W) = Capture Event 4 Interrupt Enable
3	CEVT3	R/W	0h	Capture Event 3 Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Capture Event 3 as an Interrupt source 1h (R/W) = Enable Capture Event 3 as an Interrupt source
2	CEVT2	R/W	0h	Capture Event 2 Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Capture Event 2 as an Interrupt source 1h (R/W) = Enable Capture Event 2 as an Interrupt source

**Table 19-14. ECEINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	CEVT1	R/W	0h	Capture Event 1 Interrupt Enable Reset type: SYSRStn 0h (R/W) = Disable Capture Event 1 as an Interrupt source 1h (R/W) = Enable Capture Event 1 as an Interrupt source
0	RESERVED	R	0h	Reserved

**19.9.2.11 ECFLG Register (Offset = 17h) [reset = 0h]**

ECFLG is shown in [Figure 19-27](#) and described in [Table 19-15](#).

Return to the [Summary Table](#).

Capture Interrupt Flag Register

**Figure 19-27. ECFLG Register**

15	14	13	12	11	10	9	8
RESERVED							RESERVED
R-0h							
7	6	5	4	3	2	1	0
CTR_CMP	CTR_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	INT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 19-15. ECFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7	CTR_CMP	R	0h	Compare Equal Compare Status Flag. This flag is active only in APWM mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the counter (TSCTR) reached the compare register value (ACMP)
6	CTR_PRD	R	0h	Counter Equal Period Status Flag. This flag is only active in APWM mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the counter (TSCTR) reached the period register value (APRD) and was reset.
5	CTROVF	R	0h	Counter Overflow Status Flag. This flag is active in CAP and APWM mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the counter (TSCTR) has made the transition from FFFFFFFF to 00000000
4	CEVT4	R	0h	Capture Event 4 Status Flag This flag is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the fourth event occurred at ECAPx pin
3	CEVT3	R	0h	Capture Event 3 Status Flag. This flag is active only in CAP mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the third event occurred at ECAPx pin.
2	CEVT2	R	0h	Capture Event 2 Status Flag. This flag is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the second event occurred at ECAPx pin.
1	CEVT1	R	0h	Capture Event 1 Status Flag. This flag is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the first event occurred at ECAPx pin.

**Table 19-15. ECFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	INT	R	0h	Global Interrupt Status Flag Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates that an interrupt was generated.

**19.9.2.12 ECCLR Register (Offset = 18h) [reset = 0h]**

ECCLR is shown in [Figure 19-28](#) and described in [Table 19-16](#).

Return to the [Summary Table](#).

Capture Interrupt Clear Register

**Figure 19-28. ECCLR Register**

15	14	13	12	11	10	9	8
RESERVED							RESERVED
R-0h							
7	6	5	4	3	2	1	0
CTR_CMP	CTR_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	INT
R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h

**Table 19-16. ECCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	RESERVED	R-0/W1C	0h	Reserved
7	CTR_CMP	R-0/W1C	0h	Counter Equal Compare Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CTR=CMP flag.
6	CTR_PRD	R-0/W1C	0h	Counter Equal Period Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CTR=PRD flag.
5	CTROVF	R-0/W1C	0h	Counter Overflow Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CTROVF flag.
4	CEVT4	R-0/W1C	0h	Capture Event 4 Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT4 flag.
3	CEVT3	R-0/W1C	0h	Capture Event 3 Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT3 flag.
2	CEVT2	R-0/W1C	0h	Capture Event 2 Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT2 flag.
1	CEVT1	R-0/W1C	0h	Capture Event 1 Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT1 flag.
0	INT	R-0/W1C	0h	ECAP Global Interrupt Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the INT flag and enable further interrupts to be generated if any of the event flags are set to 1

### 19.9.2.13 ECFRC Register (Offset = 19h) [reset = 0h]

ECFRC is shown in [Figure 19-29](#) and described in [Table 19-17](#).

Return to the [Summary Table](#).

Capture Interrupt Force Register

**Figure 19-29. ECFRC Register**

15	14	13	12	11	10	9	8
RESERVED							RESERVED
R-0h							
7	6	5	4	3	2	1	0
CTR_CMP	CTR_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	RESERVED
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0h

**Table 19-17. ECFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	CTR_CMP	R-0/W1S	0h	Force Counter Equal Compare Interrupt. This event is only active in APWM mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CTR_CMP flag.
6	CTR_PRD	R-0/W1S	0h	Force Counter Equal Period Interrupt. This event is only active in APWM mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CTR_PRD flag.
5	CTROVF	R-0/W1S	0h	Force Counter Overflow Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 to this bit sets the CTROVF flag.
4	CEVT4	R-0/W1S	0h	Force Capture Event 4. This event is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CEVT4 flag.
3	CEVT3	R-0/W1S	0h	Force Capture Event 3. This event is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CEVT3 flag.
2	CEVT2	R-0/W1S	0h	Force Capture Event 2. This event is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CEVT2 flag.
1	CEVT1	R-0/W1S	0h	Force Capture Event 1. This event is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Sets the CEVT1 flag.
0	RESERVED	R	0h	Reserved

### 19.9.3 Register to Driverlib Function Mapping

**Table 19-18. ECAP Registers to Driverlib Functions**

File	Driverlib Function
<b>TSCTR</b>	
ecap.h	ECAP_getTimeBaseCounter
<b>CTRPHS</b>	
ecap.h	ECAP_setPhaseShiftCount
<b>CAP1</b>	
ecap.h	ECAP_setAPWMPeriod
ecap.h	ECAP_getEventTimeStamp
<b>CAP2</b>	
ecap.h	ECAP_setAPWMCompare
ecap.h	ECAP_getEventTimeStamp
<b>CAP3</b>	
ecap.h	ECAP_setAPWMShadowPeriod
ecap.h	ECAP_getEventTimeStamp
<b>CAP4</b>	
ecap.h	ECAP_setAPWMShadowCompare
ecap.h	ECAP_getEventTimeStamp
<b>ECCTL0</b>	
ecap.h	ECAP_selectECAPInput
<b>ECCTL1</b>	
ecap.c	ECAP_setEmulationMode
ecap.h	ECAP_setEventPrescaler
ecap.h	ECAP_setEventPolarity
ecap.h	ECAP_enableCounterResetOnEvent
ecap.h	ECAP_disableCounterResetOnEvent
ecap.h	ECAP_enableTimeStampCapture
ecap.h	ECAP_disableTimeStampCapture
<b>ECCTL2</b>	
ecap.h	ECAP_setCaptureMode
ecap.h	ECAP_reArm
ecap.h	ECAP_enableCaptureMode
ecap.h	ECAP_enableAPWMMode
ecap.h	ECAP_enableLoadCounter
ecap.h	ECAP_disableLoadCounter
ecap.h	ECAP_loadCounter
ecap.h	ECAP_setSyncOutMode
ecap.h	ECAP_stopCounter
ecap.h	ECAP_startCounter
ecap.h	ECAP_setAPWMPolarity
ecap.h	ECAP_resetCounters
ecap.h	ECAP_setDMASource
ecap.h	ECAP_getModuloCounterStatus
<b>ECEINT</b>	
ecap.h	ECAP_enableInterrupt
ecap.h	ECAP_disableInterrupt
<b>ECFLG</b>	
ecap.h	ECAP_getInterruptSource



**Table 19-18. ECAP Registers to Driverlib Functions (continued)**

File	Driverlib Function
ecap.h	ECAP_getGlobalInterruptStatus
<b>ECCLR</b>	
ecap.h	ECAP_clearInterrupt
ecap.h	ECAP_clearGlobalInterrupt
<b>ECFRC</b>	
ecap.h	ECAP_forceInterrupt

## High Resolution Capture (HRCAP)

---



---

This chapter describes the operation of the high resolution capture (HRCAP) module. The HRCAP submodule described here is part of the Type-1 eCAP. HRCAP measures the width of external pulses to a higher degree of accuracy than the eCAP module. See the [TMS320x28xx, 28xxx DSP Peripheral Reference Guide](#) for a list of all devices with an HRCAP module of the same type, to determine the differences between types, and for a list of device-specific differences within a type. A detailed description of all referenced functions can be found in the C2000Ware documentation .

Topic	Page
<b>20.1 Introduction</b> .....	<b>2123</b>
<b>20.2 Description</b> .....	<b>2123</b>
<b>20.3 Operational Details</b> .....	<b>2123</b>
<b>20.4 Known Exceptions</b> .....	<b>2127</b>
<b>20.5 HRCAP Registers</b> .....	<b>2128</b>

## 20.1 Introduction

Uses for the HRCAP module include:

- Capacitive touch applications
- High-resolution period and duty cycle measurements of pulse train cycles
- Instantaneous speed measurements
- Instantaneous frequency measurements
- Voltage measurements across an isolation boundary
- Distance/sonar measurement and scanning
- Measuring flow

The HRCAP submodule includes the following features:

- Pulse-width capture in either non-high-resolution or high-resolution modes
- Absolute mode pulse-width capture
- Continuous or one-shot capture
- Interrupt on either falling or rising edge
- Continuous mode capture of pulse widths in 4-deep buffer
- Hardware calibration logic for precision high-resolution capture

All of the above resources are available on any pin using the Input X-BAR

Improvements to the Type 0 HRCAP are as follows:

- Simplified calibration scheme
  - HRCAP is always functional; never offline to perform calibration
  - Calibration is always running in the background; drastically reduced software overhead to calibrate
- Reduced software overhead to compute fractional bits
- Fractional and integer portions are packed into 32 bits
- All eCAP hardware is accessible when using the HRCAP enhancements. See [Known Exceptions](#) for practical considerations.
- Usage of the HRCAP is now unified with the eCAP

## 20.2 Description

The HRCAP enhancement has been added to eCAP 6 and eCAP 7 to allow signals to be captured asynchronously to SYSCLK. Each HRCAP submodule includes one capture channel in addition to a hardware calibration block. All eCAP hardware is accessible when using the HRCAP enhancements; however, using the Event Filter or the Input Qualifier is not valid, as these are synchronous to SYSCLK.

Each HRCAP-capable channel has the following independent key resources:

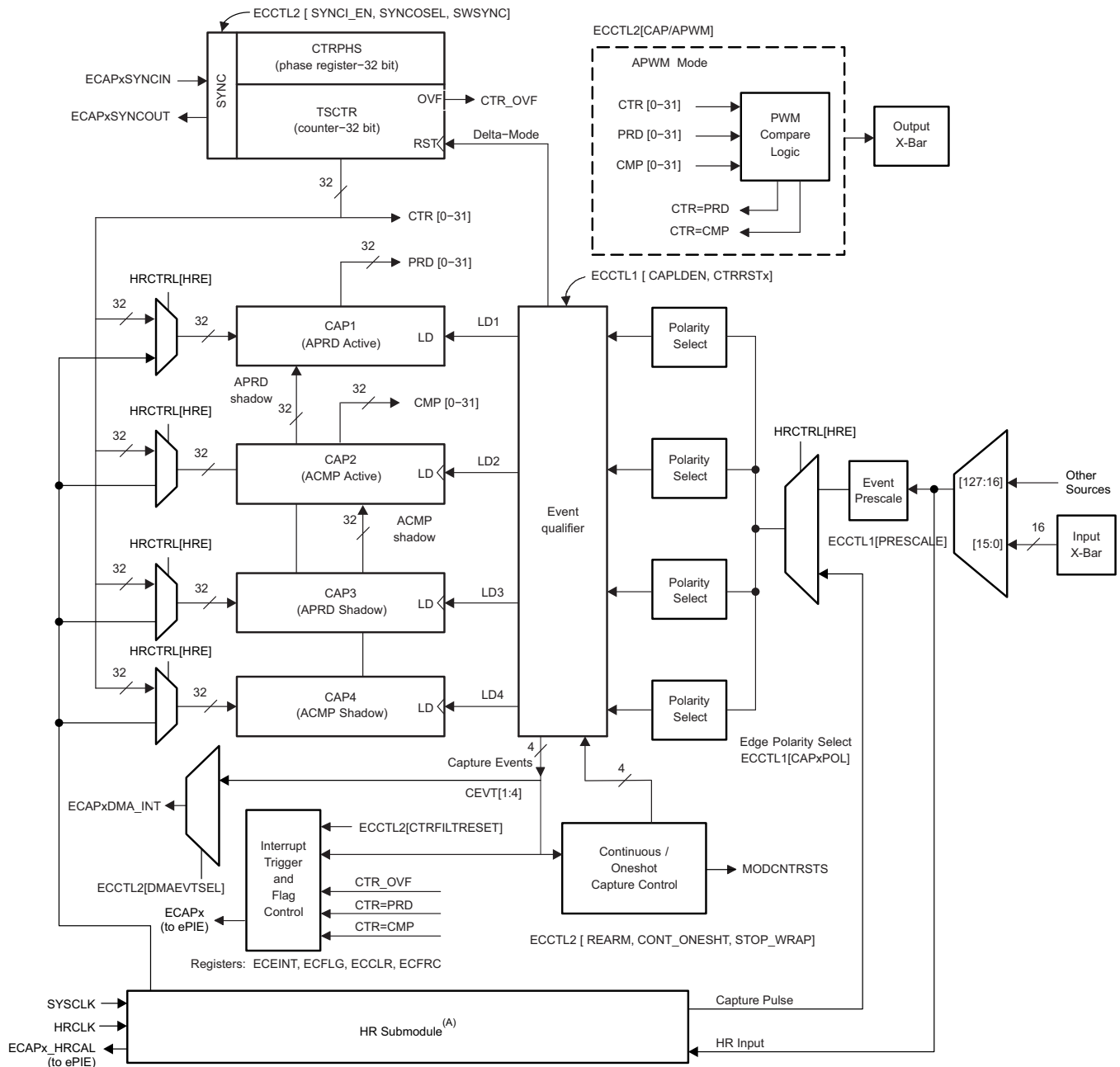
- All hardware of the respective eCAP
- High-resolution calibration logic
- Dedicated calibration interrupt

## 20.3 Operational Details

[Figure 20-1](#) shows the various components that implement the high-resolution capture functionality of the eCAP module. Note that existing eCAP resources are reused, which requires that the eCAP module is set up before using the HRCAP enhancements. For simplicity, absolute timestamp measurements are recommended. See [Known Exceptions](#) for more details.

All HRCAP measurements are relative-time measures, in terms of minimum step size. Calibration hardware as well as software functions, have been provided to convert relative-time measurements to time-converted measurements in terms of seconds. The calibration hardware and software is only required if time-converted measurements are required.

Figure 20-1. HRCAP Operations Block Diagram



Copyright © 2018, Texas Instruments Incorporated

A The HRCAP submodule is not available on all eCAP modules; in this case, the high-resolution muxes and hardware are not implemented.

### 20.3.1 HRCAP Clocking

Unlike previous Type-0 HRCAP modules, the Type-1 eCAP, with HRCAP functionality, does not require a second PLL. However, the module still requires both SYSCLK and a second asynchronous clock source called HRCLK. The HRCLK is sensitive to changes in both temperature and voltage. For this reason, when using time-converted measurements, it is required to make periodic continuous calibrations.

### 20.3.2 HRCAP Initialization Sequence

To set up the HRCAP to make time-converted measurements:

1. Enable HRCLK using `HRCAP_enableHighResolutionClock()`
2. Delay 1uS
3. Configure the eCAP module as desired, including interrupts
4. Enable HR mode using `HRCAP_enableHighResolution()`
5. Set calibration period using `HRCAP_setCalibrationPeriod()`
6. Enable continuous calibration using `HRCAP_setCalibrationMode()`
7. Enable interrupts using `HRCAP_enableCalibrationInterrupt()`
8. Start calibration using `HRCAP_startCalibration()`

Steps 5-8 only apply for time-converted measurements. When using the HRCAP to take relative-time measurements only steps 1-4 are required.

### 20.3.3 HRCAP Interrupts

The HRCAP enhancements leverage the existing eCAP interrupts (see [Section 19.6.7](#)) in addition to HRCALINT which is used exclusively by the hardware calibration block. HRCALINT can be triggered by the following conditions:

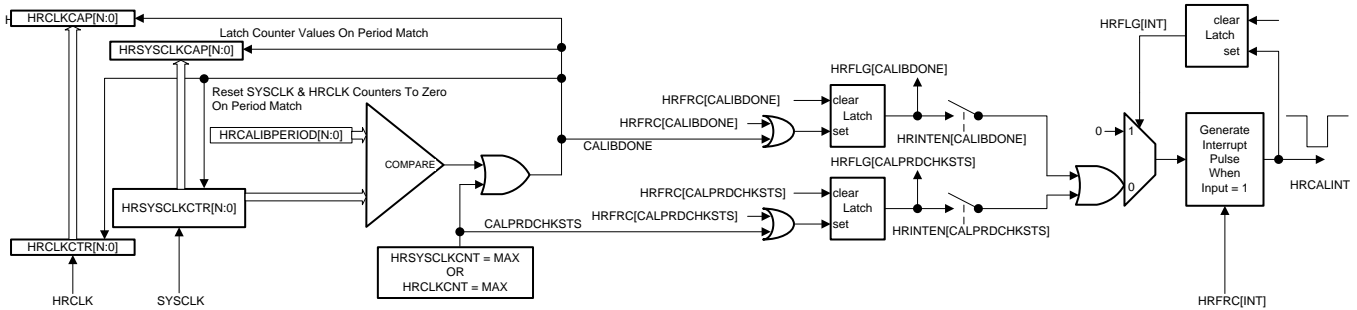
1. SYSCLKCTR = HRCALIBPERIOD
2. SYSCLKCTR or HRCLKCTR experience an overflow condition

Figure 20-2 shows this logic.

### 20.3.4 HRCAP Calibration

The following section applies only to time-converted measurements; calibration for relative-time measurements is not required. All values captured by the HRCAP submodule are in number of HRCLK cycles. The HRCLK speed varies widely with temperature and voltage, thus a scale factor is required to convert the capture value to the SYSCLK domain. For the same reason, it is required to periodically recalculate the scale factor. The HRCAP submodule has a calibration block to reduce software overhead when calculating a scale factor between HRCLK and SYSCLK.

**Figure 20-2. HRCAP Calibration**



The calibration block contains the following key resources:

- HRSYSCLKCNT
  - A 32-bit counter connected to SYSCLK. The counter will start counting when CALIBSTART is set.
- HRCLKCNT
  - A 32-bit counter connected to HRCLK. The counter will start counting when CALIBSTART is set.
- HRCALIBPERIOD
  - Calibration period, calibration is stopped when HRSYSCLKCNT is equal to the value in this register.
- HRSYSCLKCAP
  - On a calibration period match, the value of HRSYSCLKCNT is captured into HRSYSCLKCAP
- HRCLKCAP
  - On a calibration period match, the value of HRCLKCNT is captured into HRCLKCAP
- HRCALINT
  - An interrupt that occurs on a calibration period match, or when one of the counter registers experiences an overflow condition.

The calibration logic consists of two free-running counters; one clocked by HRCLK(HRCLKCTR) and the other clocked by SYSCLK(HRSYSCLKCTR). When HRSYSCLKCTR is equal to HRCALIBPERIOD, the calibration block will capture and reset both counter values, then trigger an interrupt, indicating a new scale factor is ready to be calculated. The scale factor can be found by dividing HRSYSCLKCAP by HRCLKCAP (see [Equation 1](#)). A DriverLib function, HRCAP\_getScaleFactor, has been provided to determine the scale factor. This function should be called inside of the calibration interrupt service routine. If one of the counters experiences overflow, the CALPRDCHKSTS flag will be set. The full details of the calibration block are described in [Figure 20-2](#).

$$ScaleFactor = \frac{HRSYSCLKCAP}{HRCLKCAP} \quad (2)$$

**NOTE:** Even with calibration, noise on the 1.2V VDD supply will negatively affect the standard deviation of the HRCAP submodule. Care should be taken to ensure that the 1.2V supply is clean, and that noisy internal events such as enabling and disabling clock trees have been minimized while using the HRCAP submodule.

#### 20.3.4.1 Applying the Scale Factor

A DriverLib function has been provided to apply the scale factor to a capture value, HRCAP\_getEventTimeStampNanoseconds(). [Equation 2](#) shows how to convert a raw count to seconds without using the DriverLib function.

$$Measurement(ns) = \frac{RawCount \times scaleFactor}{128} * SysClkPrd(ns) \quad (3)$$

**Table 20-1. Scale Factor**

Parameter	Typical value	Explanation
RawCount	9300	Capture value as read from ECAP_REGS_CAP1-4
ScaleFactor	2.75	The Scale factor as calculated from <a href="#">Equation 2</a>
128	128	Constant determined by the hardware of the HRCAP submodule
SysClkPrd(nS)	10	Period of the system clock
Measurement(nS)	1998.04	Signal converted to nS

### 20.3.5 DriverLib Functions

## 20.4 Known Exceptions

In HRCAP mode:

- Enabling and disabling core clocks negatively affects the standard deviation of the HRCAP submodule. Do not enable or disable core clocks while taking measurements.
- TSCTR is not writable; however, it can be reset using ECCTL2[CTRFILTRESET]
- Input synchronization is not applicable when using the HRCAP enhancements, because the HRCAP submodule is asynchronous to SYSCLK.
- The Event Filter functionality is not applicable for HRCAP, which defeats the purpose of HRCAP as the Event Filter's output is synchronous to SYSCLK.
- The best practice is to use absolute time mode for high resolution mode. If time difference mode is used, it may lead to inaccurate results if the fractional value is not taken into consideration for capture events which have reset the time base counter.
  - Actual Capture Value = (Capture Value) – (fractional value of reference event which reset the counter)
- For high frequency input signals, the CPU may not be able cope with the speed of the captures. In such a case, One-Shot mode is recommended. This mode allows the device to capture up to four edges before waiting to be serviced when the CPU is ready. This is applicable for the eCAP as well; however in that case the event filter can be used to reduce the rate of captures.

## 20.5 HRCAP Registers

This section describes the High-Resolution Capture Registers.

### 20.5.1 HRCAP Base Addresses

**Table 20-2. HRCAP Base Address Table**

Device Registers	Register Name	Start Address	End Address
Hrcap6Regs	HRCAP_REGS	0x0000_5360	0x0000_537F
Hrcap7Regs	HRCAP_REGS	0x0000_53A0	0x0000_53BF



## 20.5.2 HRCAP\_REGS Registers

Table 20-3 lists the HRCAP\_REGS registers. All register offset addresses not listed in Table 20-3 should be considered as reserved locations and the register contents should not be modified.

**Table 20-3. HRCAP\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	HRCTL	High-Res Control Register	EALLOW	<a href="#">Go</a>
4h	HRINTEN	High-Res Calibration Interrupt Enable Register	EALLOW	<a href="#">Go</a>
6h	HRFLG	High-Res Calibration Interrupt Flag Register		<a href="#">Go</a>
8h	HRCLR	High-Res Calibration Interrupt Clear Register		<a href="#">Go</a>
Ah	HRFRC	High-Res Calibration Interrupt Force Register	EALLOW	<a href="#">Go</a>
Ch	HRCALPRD	High-Res Calibration Period Register	EALLOW	<a href="#">Go</a>
Eh	HRSYSCLKCTR	High-Res Calibration SYSCLK Counter Register		<a href="#">Go</a>
10h	HRSYSCLKCAP	High-Res Calibration SYSCLK Capture Register		<a href="#">Go</a>
12h	HRCLKCTR	High-Res Calibration HRCLK Counter Register		<a href="#">Go</a>
14h	HRCLKCAP	High-Res Calibration HRCLK Capture Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 20-4 shows the codes that are used for access types in this section.

**Table 20-4. HRCAP\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1C	W1C	Write 1 to clear
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 20.5.2.1 HRCTL Register (Offset = 0h) [reset = 0h]

HRCTL is shown in [Figure 20-3](#) and described in [Table 20-5](#).

Return to the [Summary Table](#).

High-Res Control Register

**Figure 20-3. HRCTL 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		CALIBCONT	CALIBSTS	CALIBSTART	PRDSEL	HRCLKE	HRE
R/W-0h		R/W-0h	R-0-0h	R-0/W1S-0h	R/W-0h	R/W-0h	R/W-0h

**Table 20-5. HRCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R/W	0h	Reserved
5	CALIBCONT	R/W	0h	Continuous mode Calibration Select Bit: 0 Continuous mode disabled. 1 Continuous mode enabled. Calibration automatically restarts at end of current calibration cycle. Reset type: CPU1.SYSRSn
4	CALIBSTS	R-0	0h	Calibration status Bit: 0 No active calibration cycle 1 Calibration cycle in progress Reset type: CPU1.SYSRSn
3	CALIBSTART	R-0/W1S	0h	Calibration start Bit: 0 No effect 1 Starts the calibration cycle Reset type: CPU1.SYSRSn
2	PRDSEL	R/W	0h	Calibration Period Match Select Bit: 0 Use SYSCLK Counter For Period Match (default at reset) 1 Reserved Reset type: CPU1.SYSRSn
1	HRCLKE	R/W	0h	High Resolution Clock Enable Bit: 0 High resolution clock disabled (default at reset) 1 High resolution clock enabled. The clock should be enabled before enabling the high res function via the HRE bit. Reset type: CPU1.SYSRSn

**Table 20-5. HRCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	HRE	R/W	0h	<p>High Resolution Enable Bit:</p> <p>0 High resolution mode disabled (default at reset)</p> <p>1 High resolution mode enabled. Enabling this mode will connect the capture registers and edge event modes of the ECAP to be accessed by the High Res function.</p> <p>Note: The High Res clock needs to be enabled (using the HRCLKE bit) first before enabling the module. Allow a certain start up stabilization period before enabling the module.</p> <p>Reset type: CPU1.SYSRSn</p>

### 20.5.2.2 HRINTEN Register (Offset = 4h) [reset = 0h]

HRINTEN is shown in [Figure 20-4](#) and described in [Table 20-6](#).

Return to the [Summary Table](#).

High-Res Calibration Interrupt Enable Register

**Figure 20-4. HRINTEN Register**

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

**Table 20-6. HRINTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2	CALPRDCHKSTS	R/W	0h	Calibration Period Check status Interrupt Enable: 0 Disable Calibration Period Check interrupt status 1 Enable Calibration Period Check interrupt status Reset type: CPU1.SYSRSn
1	CALIBDONE	R/W	0h	Calibration done Interrupt Enable: 0 Disable Calibration done Interrupt 1 Enable Calibration done Interrupt Reset type: CPU1.SYSRSn
0	RESERVED	R-0	0h	Reserved

**20.5.2.3 HRFLG Register (Offset = 6h) [reset = 0h]**

HRFLG is shown in [Figure 20-5](#) and described in [Table 20-7](#).

Return to the [Summary Table](#).

High-Res Calibration Interrupt Flag Register

**Figure 20-5. HRFLG Register**

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

**Table 20-7. HRFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2	CALPRDCHKSTS	R	0h	Calibration period check status Flag Bit: 1 Indicates that calibration ended before PRDCHK due to overflow on one of the counters. 0 Indicates no event occurred.  Note: This bit remains latched until cleared by the user using the HRCLR [CALPRDCHKSTS] bit. Reset type: CPU1.SYSRSn
1	CALIBDONE	R	0h	Calibration Done Interrupt Flag Bit: 1 Indicates calibration cycle is completed 0 Indicates calibration cycle has not completed.  Note: This bit remains latched until cleared by the user using the HRCLR [CALIBDONE] bit. Reset type: CPU1.SYSRSn
0	CALIBINT	R	0h	Global calibration Interrupt Status Flag: 1 Indicates that an interrupt was generated from CALIBDONE or CALPRDCHKSTS. 0 Indicates no interrupt generated. Reset type: CPU1.SYSRSn

### 20.5.2.4 HRCLR Register (Offset = 8h) [reset = 0h]

HRCLR is shown in [Figure 20-6](#) and described in [Table 20-8](#).

Return to the [Summary Table](#).

High-Res Calibration Interrupt Clear Register

**Figure 20-6. HRCLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED					CALPRDCHKSTS	CALIBDONE	CALIBINT
R-0-0h					R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h

**Table 20-8. HRCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2	CALPRDCHKSTS	R-0/W1C	0h	Clear Calibration period check status Flag Bit: 1 Clears the CALPRDCHKSTS flag register bit. 0 No effect.  Note: H/W has priority over CPU writes if the user tries to clear a flag bit and an event occurs on the same cycle that tries to set the flag for the selected bit. Reset type: CPU1.SYSRSn
1	CALIBDONE	R-0/W1C	0h	Clear Calibration Done Interrupt Flag Bit: 1 Clears the CALIBDONE interrupt flag register bit. 0 No effect.  Note: H/W has priority over CPU writes if the user tries to clear a flag bit and an event occurs on the same cycle that tries to set the flag for the selected bit. Reset type: CPU1.SYSRSn
0	CALIBINT	R-0/W1C	0h	Clear Global calibration Interrupt Flag 1 Clears the Global interrupt flag and enables further interrupts to be generated if any of the event flags are set. 0 No effect. Reset type: CPU1.SYSRSn

### 20.5.2.5 HRFRC Register (Offset = Ah) [reset = 0h]

HRFRC is shown in [Figure 20-7](#) and described in [Table 20-9](#).

Return to the [Summary Table](#).

High-Res Calibration Interrupt Force Register

**Figure 20-7. HRFRC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED					CALPRDCHKSTS	CALIBDONE	RESERVED
R-0-0h					R-0/W1S-0h	R-0/W1S-0h	R-0-0h

**Table 20-9. HRFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2	CALPRDCHKSTS	R-0/W1S	0h	Force CALPRDCHKSTS flag: 0 No effect 1 Sets the CALPRDCHKSTS flag. Reset type: CPU1.SYSRSn
1	CALIBDONE	R-0/W1S	0h	Force CALIBDONE flag: 0 No effect 1 Sets the CALIBDONE flag. Reset type: CPU1.SYSRSn
0	RESERVED	R-0	0h	Reserved

### 20.5.2.6 HRCALPRD Register (Offset = Ch) [reset = 003FFFFFFh]

HRCALPRD is shown in [Figure 20-8](#) and described in [Table 20-10](#).

Return to the [Summary Table](#).

High-Res Calibration Period Register

**Figure 20-8. HRCALPRD 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
PRD																															
R/W-003FFFFFFh																															

**Table 20-10. HRCALPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	PRD	R/W	003FFFFFFh	Register to program calibration period. The period value is matched against HRSYSCLKCTR. On a match an interrupt is generated and the counter registers values are captured. Reset type: CPU1.SYSRSn



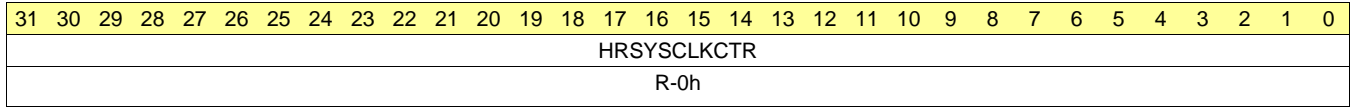
**20.5.2.7 HRSYSCLKCTR Register (Offset = Eh) [reset = 0h]**

HRSYSCLKCTR is shown in [Figure 20-9](#) and described in [Table 20-11](#).

Return to the [Summary Table](#).

High-Res Calibration SYSCLK Counter Register

**Figure 20-9. HRSYSCLKCTR Register**



**Table 20-11. HRSYSCLKCTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	HRSYSCLKCTR	R	0h	Current SYSCLK counter value Reset type: CPU1.SYSRSn

### 20.5.2.8 HRSYSCLKCAP Register (Offset = 10h) [reset = 0h]

HRSYSCLKCAP is shown in [Figure 20-10](#) and described in [Table 20-12](#).

Return to the [Summary Table](#).

High-Res Calibration SYSCLK Capture Register

**Figure 20-10. HRSYSCLKCAP 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
HRSYSCLKCAP																															
R-0h																															

**Table 20-12. HRSYSCLKCAP Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	HRSYSCLKCAP	R	0h	HRSYSCLKCAP captures into this register at end of calibration cycle. Reset type: CPU1.SYSRSn

**20.5.2.9 HRCLKCTR Register (Offset = 12h) [reset = 0h]**

HRCLKCTR is shown in [Figure 20-11](#) and described in [Table 20-13](#).

Return to the [Summary Table](#).

High-Res Calibration HRCLK Counter Register

**Figure 20-11. HRCLKCTR 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
HRCLKCTR																															
R-0h																															

**Table 20-13. HRCLKCTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	HRCLKCTR	R	0h	Current HRCLK counter value Reset type: CPU1.SYSRSn

**20.5.2.10 HRCLKCAP Register (Offset = 14h) [reset = 0h]**

HRCLKCAP is shown in [Figure 20-12](#) and described in [Table 20-14](#).

Return to the [Summary Table](#).

High-Res Calibration HRCLK Capture Register

**Figure 20-12. HRCLKCAP 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
HRCLKCAP																															
R-0h																															

**Table 20-14. HRCLKCAP Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	HRCLKCAP	R	0h	HRCLKCAP captures into this register at end of calibration cycle. Reset type: CPU1.SYSRSn

## **Enhanced Quadrature Encoder Pulse (eQEP)**

---



---

The enhanced Quadrature Encoder Pulse (eQEP) module described here is a Type-1 eQEP. See the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide (SPRU566)* for a list of all devices with a module of the same type to determine the differences between types and for a list of device-specific differences within a type.

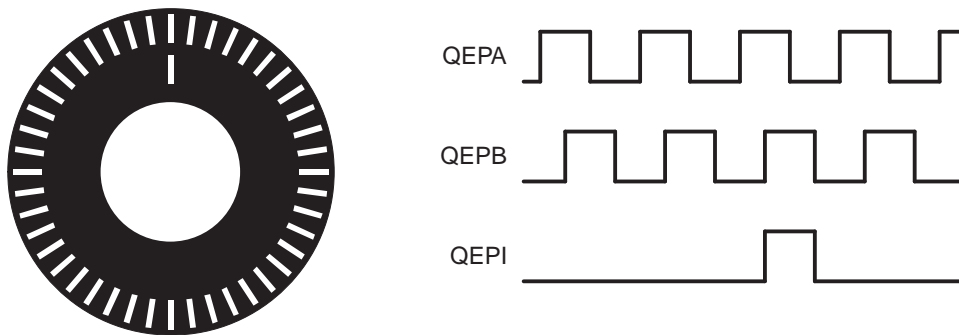
The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system.

Topic	Page
<b>21.1 Introduction .....</b>	<b>2142</b>
<b>21.2 Configuring Device Pins .....</b>	<b>2144</b>
<b>21.3 Description .....</b>	<b>2144</b>
<b>21.4 Quadrature Decoder Unit (QDU).....</b>	<b>2147</b>
<b>21.5 Position Counter and Control Unit (PCCU).....</b>	<b>2150</b>
<b>21.6 eQEP Edge Capture Unit.....</b>	<b>2156</b>
<b>21.7 eQEP Watchdog.....</b>	<b>2160</b>
<b>21.8 Unit Timer Base .....</b>	<b>2160</b>
<b>21.9 QMA Module.....</b>	<b>2161</b>
<b>21.10 eQEP Interrupt Structure .....</b>	<b>2165</b>
<b>21.11 eQEP Registers .....</b>	<b>2167</b>

## 21.1 Introduction

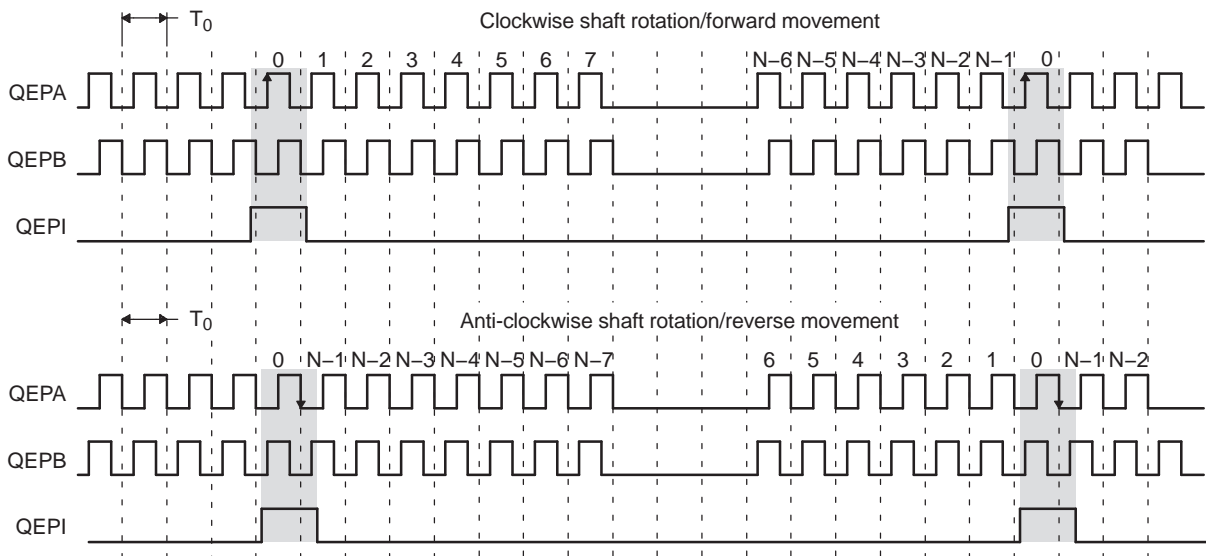
An incremental encoder disk is patterned with a track of slots along its periphery, as shown in [Figure 21-1](#). These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark and light line pairs that occur per revolution (lines per revolution). As a rule, a second track is added to generate a signal that occurs once per revolution (index signal: QEPI), which can be used to indicate an absolute position. Encoder manufacturers identify the index pulse using different terms such as index, marker, home position, and zero reference

**Figure 21-1. Optical Encoder Disk**



To derive direction information, the lines on the disk are read out by two different photo-elements that "look" at the disk pattern with a mechanical shift of 1/4 the pitch of a line pair between them. This shift is detected with a reticle or mask that restricts the view of the photo-element to the desired part of the disk lines. As the disk rotates, the two photo-elements generate signals that are shifted 90° out of phase from each other. These are commonly called the quadrature QEPA and QEPB signals. The clockwise direction for most encoders is defined as the QEPA channel going positive before the QEPB channel and vice versa as shown in [Figure 21-2](#).

**Figure 21-2. QEP Encoder Output Signal for Forward/Reverse Movement**

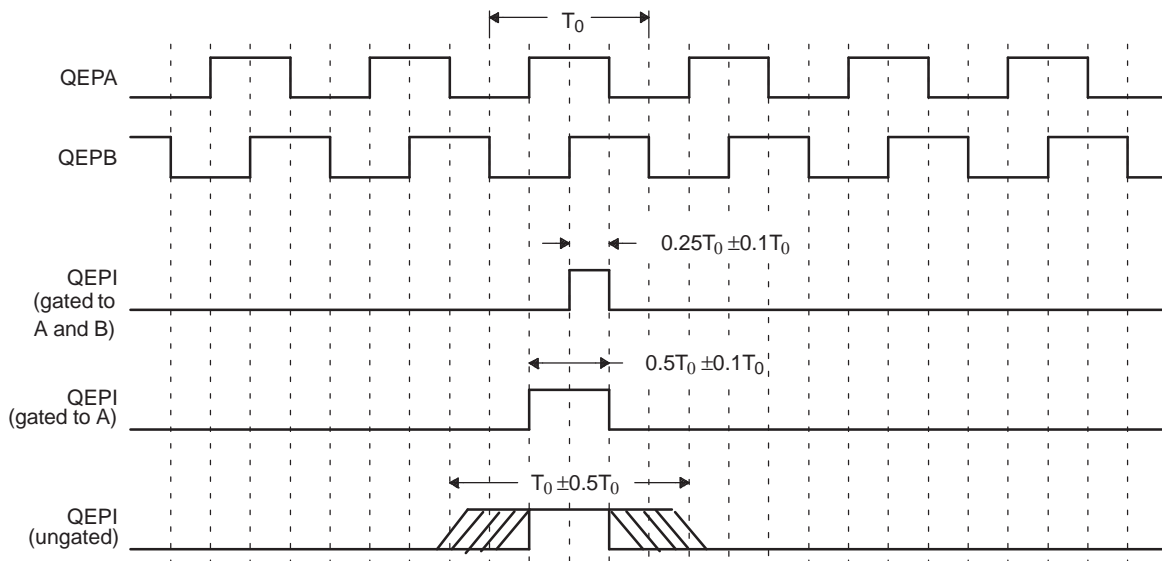


**Legend:** N = lines per revolution

The encoder wheel typically makes one revolution for every revolution of the motor, or the wheel may be at a geared rotation ratio with respect to the motor. Therefore, the frequency of the digital signal coming from the QEPA and QEPB outputs varies proportionally with the velocity of the motor. For example, a 2000-line encoder directly coupled to a motor running at 5000 revolutions per minute (rpm) results in a frequency of 166.6 KHz, so by measuring the frequency of either the QEPA or QEPB output, the processor can determine the velocity of the motor.

Quadrature encoders from different manufacturers come with two forms of index pulse (gated index pulse or ungated index pulse) as shown in Figure 21-3. A nonstandard form of index pulse is ungated. In the ungated configuration, the index edges are not necessarily coincident with A and B signals. The gated index pulse is aligned to any of the four quadrature edges and width of the index pulse and can be equal to a quarter, half, or full period of the quadrature signal.

Figure 21-3. Index Pulse Example



Some typical applications of shaft encoders include robotics and computer input in the form of a mouse. Inside your mouse you can see where the mouse ball spins a pair of axles (a left/right, and an up/down axle). These axles are connected to optical shaft encoders that effectively tell the computer how fast and in what direction the mouse is moving.

**General Issues:** Estimating velocity from a digital position sensor is a cost-effective strategy in motor control. Two different first order approximations for velocity may be written as:

$$v(k) \approx \frac{x(k) - x(k - 1)}{T} = \frac{\Delta X}{T} \quad (4)$$

$$v(k) \approx \frac{X}{t(k) - t(k - 1)} = \frac{X}{\Delta T} \quad (5)$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k-1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

t(k): Time instant "k"

t(k-1): Time instant "k-1"

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement.

Equation 4 is the conventional approach to velocity estimation and it requires a time base to provide a unit time event for velocity calculation. Unit time is basically the inverse of the velocity calculation rate.

The encoder count (position) is read once during each unit time event. The quantity  $[x(k) - x(k-1)]$  is formed by subtracting the previous reading from the current reading. Then the velocity estimate is computed by multiplying by the known constant  $1/T$  (where  $T$  is the constant time between unit time events and is known in advance).

Estimation based on [Equation 4](#) has an inherent accuracy limit directly related to the resolution of the position sensor and the unit time period  $T$ . For example, consider a 500-line per revolution quadrature encoder with a velocity calculation rate of 400 Hz. When used for position, the quadrature encoder gives a four-fold increase in resolution; in this case, 2000 counts per revolution. The minimum rotation that can be detected is therefore 0.0005 revolutions, which gives a velocity resolution of 12 rpm when sampled at 400 Hz. While this resolution may be satisfactory at moderate or high speeds, for example 1% error at 1200 rpm, it would clearly prove inadequate at low speeds. In fact, at speeds below 12 rpm, the speed estimate would erroneously be zero much of the time.

At low speed, [Equation 5](#) provides a more accurate approach. It requires a position sensor that outputs a fixed interval pulse train, such as the aforementioned quadrature encoder. The width of each pulse is defined by motor speed for a given sensor resolution. [Equation 5](#) can be used to calculate motor speed by measuring the elapsed time between successive quadrature pulse edges. However, this method suffers from the opposite limitation, as does [Equation 4](#). A combination of relatively large motor speeds and high sensor resolution makes the time interval  $\Delta T$  small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use [Equation 5](#) at low speed and have the DSP software switch over to [Equation 4](#) when the motor speed rises above some specified threshold.

## 21.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

For proper operation of the eQEP module, input GPIO pins must be configured via the GPxQSELn registers for synchronous input mode (with or without qualification). The asynchronous mode should not be used for eQEP input pins. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

## 21.3 Description

This section provides the eQEP inputs, memory map, and functional description.

### 21.3.1 EQEP Inputs

The eQEP inputs include two pins for quadrature-clock mode or direction-count mode, an index (or 0 marker), and a strobe input. The eQEP module requires that the QEPA, QEPB, and QEPI inputs are synchronized to SYSCLK prior to entering the module. The application code should enable the synchronous GPIO input feature on any eQEP-enabled GPIO pins (see the *General-Purpose Input/Output (GPIO)* chapter for more details).

- *QEPA/XCLK and QEPB/XDIR*

These two pins can be used in quadrature-clock mode or direction-count mode.

- *Quadrature-clock Mode*

The eQEP encoders provide two square wave signals (A and B) 90 electrical degrees out of phase. This phase relationship is used to determine the direction of rotation of the input shaft and number of eQEP pulses from the index position to derive the relative position information. For forward or clockwise rotation, QEPA signal leads QEPB signal and vice versa. The quadrature decoder uses these two inputs to generate quadrature-clock and direction signals.

- *Direction-count Mode*

In direction-count mode, direction and clock signals are provided directly from the external source. Some position encoders have this type of output instead of quadrature output. The QEPA pin provides the clock input and the QEPB pin provides the direction input.



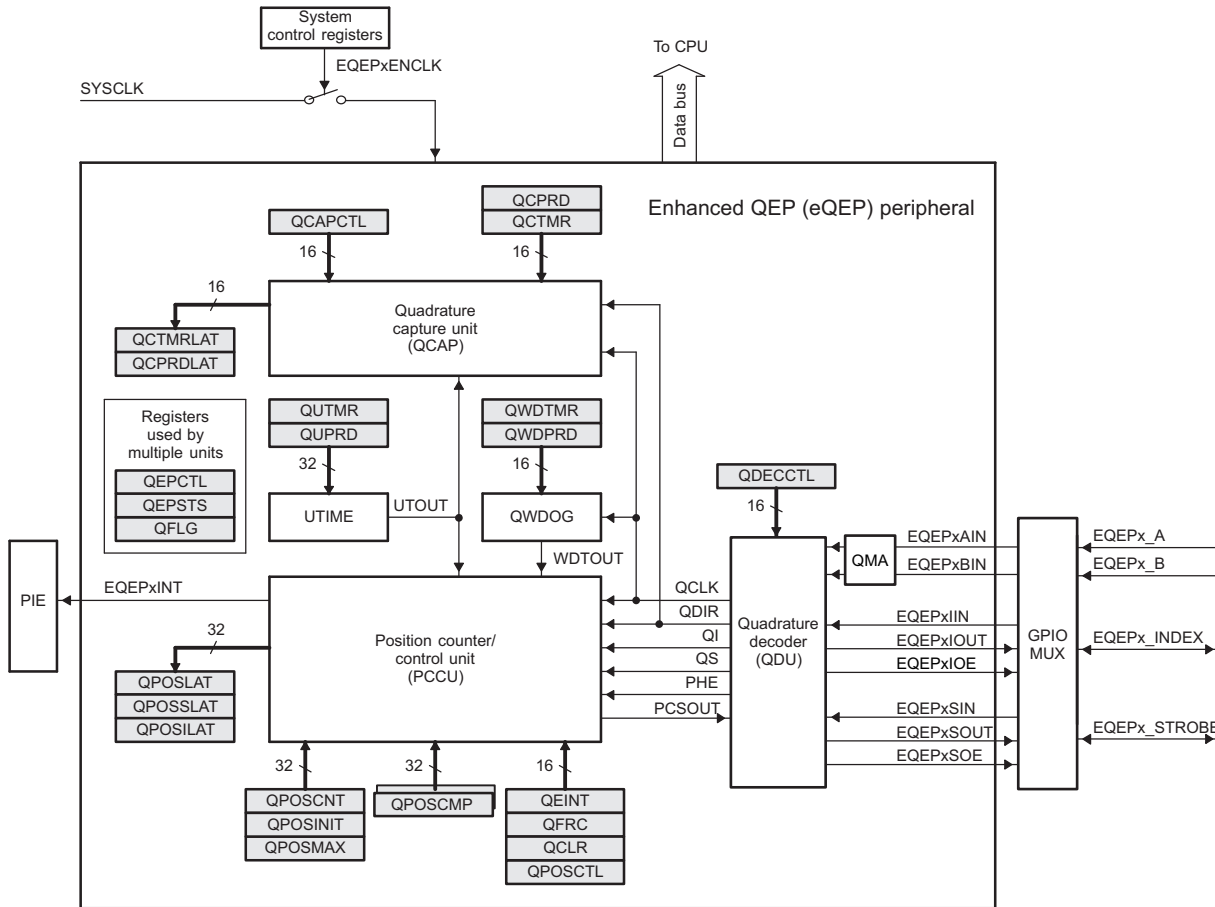
- QEPI: Index or Zero Marker**  
 The eQEP encoder uses an index signal to assign an absolute start position from which position information is incrementally encoded using quadrature pulses. This pin is connected to the index output of the eQEP encoder to optionally reset the position counter for each revolution. This signal can be used to initialize or latch the position counter on the occurrence of a desired event on the index pin.
- QEPS: Strobe Input**  
 This general-purpose strobe signal can initialize or latch the position counter on the occurrence of a desired event on the strobe pin. This signal is typically connected to a sensor or limit switch to notify that the motor has reached a defined position.

### 21.3.2 Functional Description

The eQEP peripheral contains the following major functional units (as shown in Figure 21-4):

- Programmable input qualification for each pin (part of the GPIO MUX)
- Quadrature decoder unit (QDU)
- Position counter and control unit for position measurement (PCCU)
- Quadrature edge-capture unit for low-speed measurement (QCAP)
- Unit time base for speed/frequency measurement (UTIME)
- Watchdog timer for detecting stalls (QWDOG)
- Quadrature Mode Adapter (QMA)

**Figure 21-4. Functional Block Diagram of the eQEP Peripheral**



Copyright © 2017, Texas Instruments Incorporated

### 21.3.3 eQEP Memory Map

Table 21-1 lists the registers with their memory locations, sizes, and reset values.

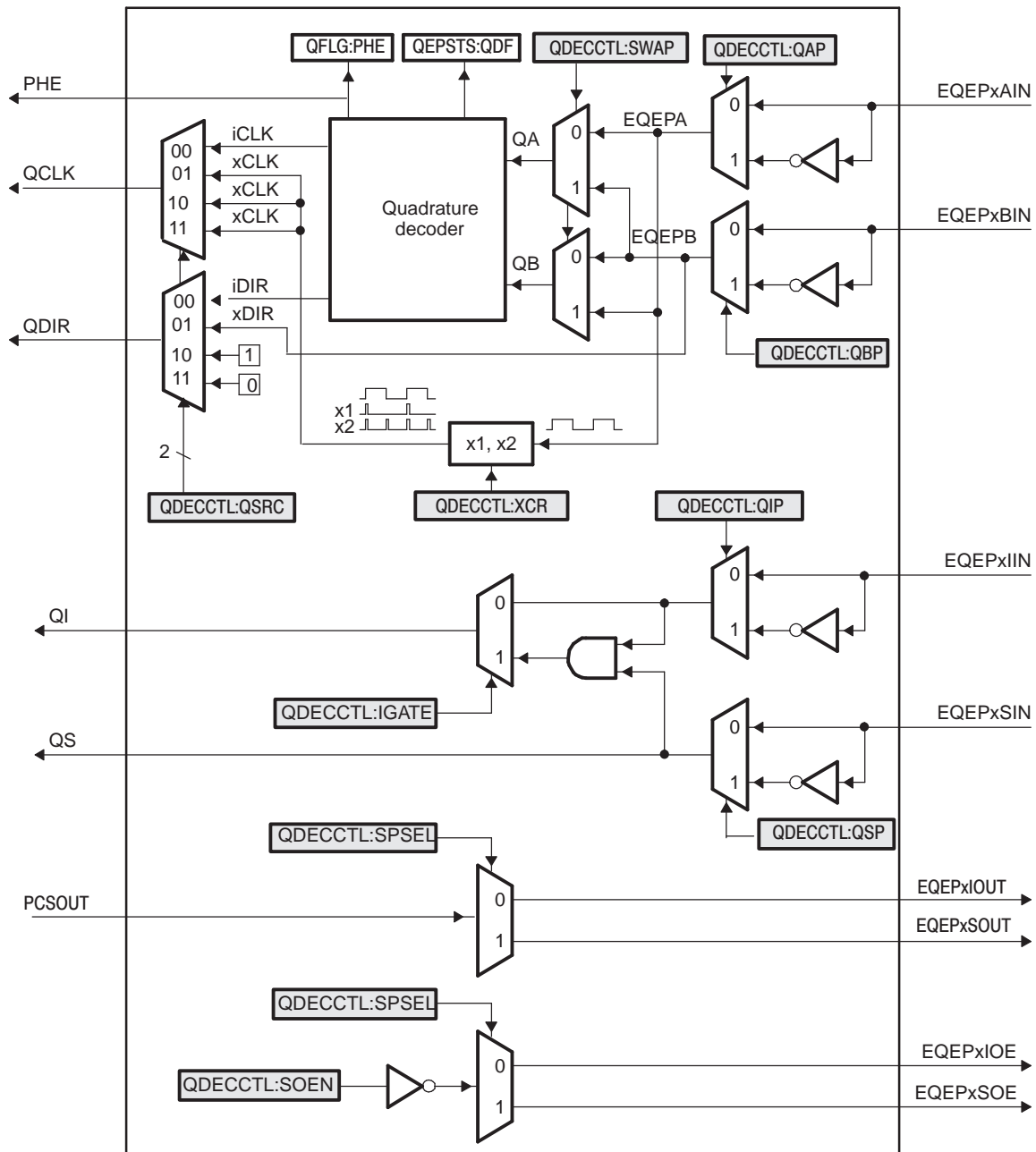
**Table 21-1. EQEP Memory Map**

Name	Offset	Size(x16)/ #shadow	Reset	Register Description
QPOSCNT	0x00	2/0	0x00000000	eQEP Position Counter
QPOSINIT	0x02	2/0	0x00000000	eQEP Initialization Position Count
QPOSMAX	0x04	2/0	0x00000000	eQEP Maximum Position Count
QPOSCMP	0x06	2/1	0x00000000	eQEP Position-compare
QPOSILAT	0x08	2/0	0x00000000	eQEP Index Position Latch
QPOSSLAT	0x0A	2/0	0x00000000	eQEP Strobe Position Latch
QPOSLAT	0x0C	2/0	0x00000000	eQEP Position Latch
QUTMR	0x0E	2/0	0x00000000	QEP Unit Timer
QUPRD	0x10	2/0	0x00000000	eQEP Unit Period Register
QWDTMR	0x12	1/0	0x0000	eQEP Watchdog Timer
QWDPRD	0x13	1/0	0x0000	eQEP Watchdog Period Register
QDECCTL	0x14	1/0	0x0000	eQEP Decoder Control Register
QEPCTL	0x15	1/0	0x0000	eQEP Control Register
QCAPCTL	0x16	1/0	0x0000	eQEP Capture Control Register
QPOSCTL	0x17	1/0	0x00000	eQEP Position-compare Control Register
QEINT	0x18	1/0	0x0000	eQEP Interrupt Enable Register
QFLG	0x19	1/0	0x0000	eQEP Interrupt Flag Register
QCLR	0x1A	1/0	0x0000	eQEP Interrupt Clear Register
QFRC	0x1B	1/0	0x0000	eQEP Interrupt Force Register
QEPSTS	0x1C	1/0	0x0000	eQEP Status Register
QCTMR	0x1D	1/0	0x0000	eQEP Capture Timer
QCPRD	0x1E	1/0	0x0000	eQEP Capture Period Register
QCTMRLAT	0x1F	1/0	0x0000	eQEP Capture Timer Latch
QCPRDLAT	0x20	1/0	0x0000	eQEP Capture Period Latch
reserved	0x21 to 0x2F	15/0		
REV	0x30	2/0	0x0000	eQEP Revision Number
QEPSTROBESEL	0x32	2/0	0x0000	eQEP Strobe select register
QMACTRL	0x34	2/0	0x0000	eQEP QMA Control register
reserved	0x36 to 0x3F	10/0		

## 21.4 Quadrature Decoder Unit (QDU)

Figure 21-5 shows a functional block diagram of the QDU.

Figure 21-5. Functional Block Diagram of Decoder Unit



### 21.4.1 Position Counter Input Modes

Clock and direction input to the position counter is selected using QDECCTL[QSRC] bits, based on interface input requirement as follows:

- Quadrature-count mode
- Direction-count mode
- UP-count mode
- DOWN-count mode

### 21.4.1.1 Quadrature Count Mode

The quadrature decoder generates the direction and clock to the position counter in quadrature count mode.

**Direction Decoding**— The direction decoding logic of the eQEP circuit determines which one of the sequences (QEPA, QEPB) is the leading sequence and accordingly updates the direction information in the QEPSTS[QDF] bit. Table 21-2 and Figure 21-6 show the direction decoding logic in truth table and state machine form. Both edges of the QEPA and QEPB signals are sensed to generate count pulses for the position counter. Therefore, the frequency of the clock generated by the eQEP logic is four times that of each input sequence. Figure 21-7 shows the direction decoding and clock generation from the eQEP input signals.

**Table 21-2. Quadrature Decoder Truth Table**

Previous Edge	Present Edge	QDIR	QPOSCNT
QA↑	QB↑	UP	Increment
	QB↓	DOWN	Decrement
	QA↓	TOGGLE	Increment or Decrement
QA↓	QB↓	UP	Increment
	QB↑	DOWN	Decrement
	QA↑	TOGGLE	Increment or Decrement
QB↑	QA↑	DOWN	Increment
	QA↓	UP	Decrement
	QB↓	TOGGLE	Increment or Decrement
QB↓	QA↓	DOWN	Increment
	QA↑	UP	Decrement
	QB↑	TOGGLE	Increment or Decrement

**Figure 21-6. Quadrature Decoder State Machine**

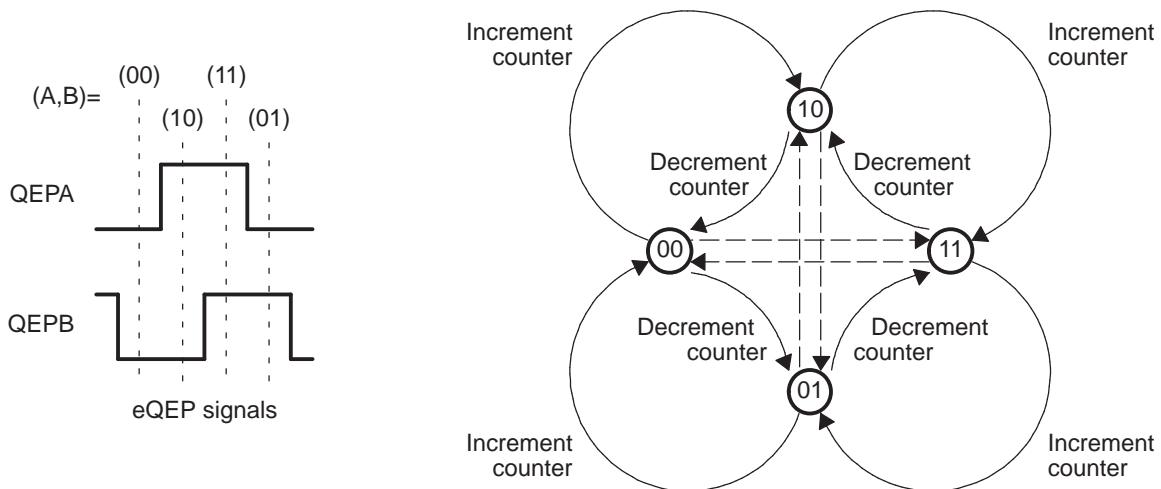
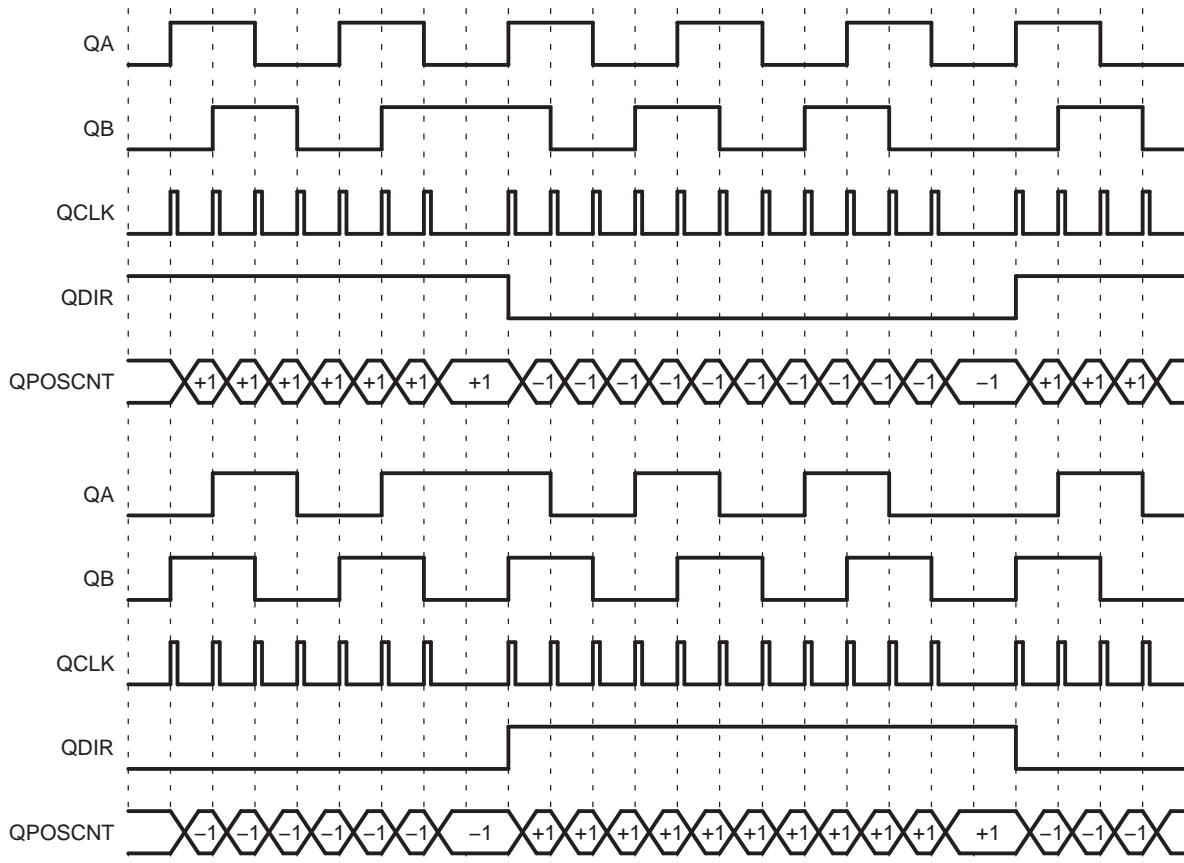


Figure 21-7. Quadrature-clock and Direction Decoding



**Phase Error Flag**— In normal operating conditions, quadrature inputs QEPA and QEPB will be 90 degrees out of phase. The phase error flag (PHE) is set in the QFLG register and the QPOSCNT value can be incorrect and offset by multiples of 1 or 3. That is, when edge transition is detected simultaneously on the QEPA and QEPB signals to optionally generate interrupts. State transitions marked by dashed lines in Figure 21-6 are invalid transitions that generate a phase error.

**Count Multiplication**— The eQEP position counter provides 4x times the resolution of an input clock by generating a quadrature-clock (QCLK) on the rising/falling edges of both eQEP input clocks (QEPA and QEPB) as shown in Figure 21-7 .

**Reverse Count**— In normal quadrature count operation, QEPA input is fed to the QA input of the quadrature decoder and the QEPB input is fed to the QB input of the quadrature decoder. Reverse counting is enabled by setting the SWAP bit in the QDECCTL register. This will swap the input to the quadrature decoder, thereby reversing the counting direction.

### 21.4.1.2 Direction-Count Mode

Some position encoders provide direction and clock outputs, instead of quadrature outputs. In such cases, direction-count mode can be used. QEPA input will provide the clock for the position counter and the QEPB input will have the direction information. The position counter is incremented on every rising edge of a QEPA input when the direction input is high, and decremented when the direction input is low.

### 21.4.1.3 Up-Count Mode

The counter direction signal is hard-wired for up-count and the position counter is used to measure the frequency of the QEPA input. Clearing the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of the QEPA input, thereby increasing the measurement resolution by a factor of 2x. In up-count mode, it is recommended that the application not configure QEPB as a GPIO mux option, or ensure that a signal edge is not generated on the QEPB input.

### 21.4.1.4 Down-Count Mode

The counter direction signal is hardwired for a down-count and the position counter is used to measure the frequency of the QEPA input. Setting the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of a QEPA input, thereby increasing the measurement resolution by a factor of 2x. In down-count mode, it is recommended that the application not configure QEPB as a GPIO mux option, or ensure that a signal edge is not generated on the QEPB input.

## 21.4.2 eQEP Input Polarity Selection

Each eQEP input can be inverted using QDECCTL[8:5] control bits. As an example, setting the QDECCTL[QIP] bit will invert the index input.

### 21.4.3 Position-Compare Sync Output

The enhanced eQEP peripheral includes a position-compare unit that is used to generate the position-compare sync signal on compare match between the position-counter register (QPOSCNT) and the position-compare register (QPOSCMP). This sync signal can be output using an index pin or strobe pin of the EQEP peripheral.

Setting the QDECCTL[SOEN] bit enables the position-compare sync output and the QDECCTL[SPSEL] bit selects either an eQEP index pin or an eQEP strobe pin.

## 21.5 Position Counter and Control Unit (PCCU)

The position-counter and control unit provides two configuration registers (QEPCTL and QPOSCTL) for setting up position-counter operational modes, position-counter initialization/latch modes and position-compare logic for sync signal generation.

### 21.5.1 Position Counter Operating Modes

Position-counter data may be captured in different manners. In some systems, the position counter is accumulated continuously for multiple revolutions and the position-counter value provides the position information with respect to the known reference. An example of this is the quadrature encoder mounted on the motor controlling the print head in the printer. Here the position counter is reset by moving the print head to the home position and then the position counter provides absolute position information with respect to home position.

In other systems, the position counter is reset on every revolution using index pulse, and the position counter provides a rotor angle with respect to the index pulse position.

The position counter can be configured to operate in following four modes

- Position-Counter Reset on Index Event
- Position-Counter Reset on Maximum Position
- Position-Counter Reset on the first Index Event
- Position-Counter Reset on Unit Time Out Event (Frequency Measurement)

In all the above operating modes, the position counter is reset to 0 on overflow and to the QPOSMAX register value on underflow. Overflow occurs when the position counter counts up after the QPOSMAX value. Underflow occurs when the position counter counts down after "0". The Interrupt flag is set to indicate overflow/underflow in QFLG register.

### 21.5.1.1 Position Counter Reset on Index Event (QEPCTL[PCRM]=00)

If the index event occurs during the forward movement, then the position counter is reset to 0 on the next eQEP. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock.

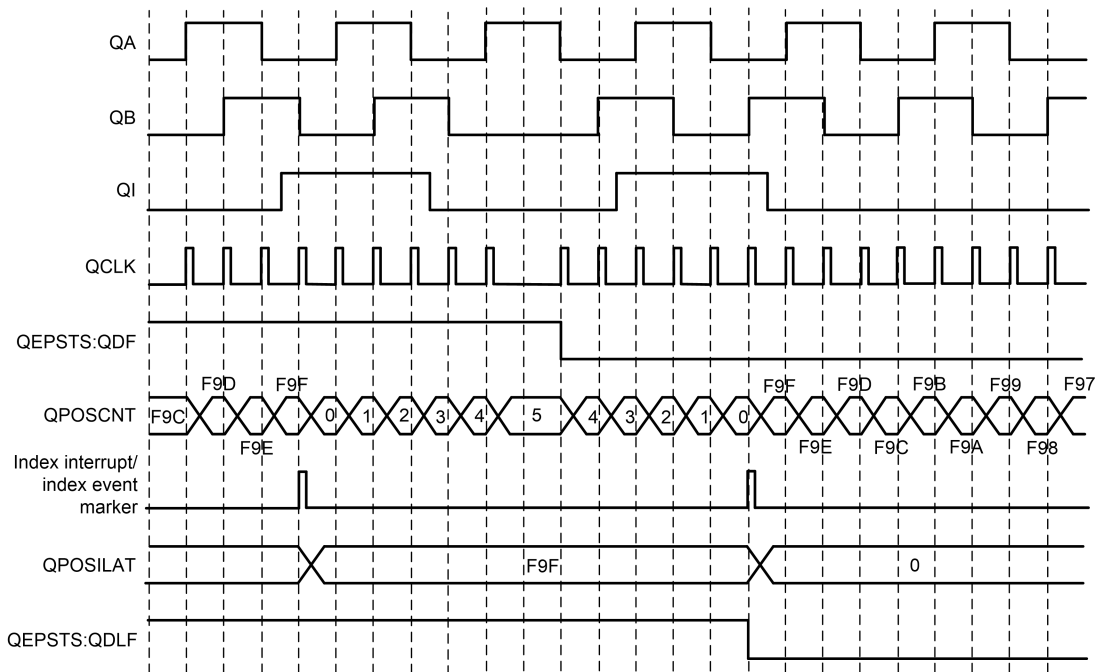
First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation.

For example, if the first reset operation occurs on the falling edge of QEPB during the forward direction, then all the subsequent reset must be aligned with the falling edge of QEPB for the forward rotation and on the rising edge of QEPB for the reverse rotation as shown in Figure 21-8.

The position-counter value is latched to the QPOSILAT register and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker. The position-counter error flag (QEPSTS[PCEF]) and error interrupt flag (QLFG[PCE]) are set if the latched value is not equal to 0 or QPOSMAX. The position-counter error flag (QEPSTS[PCEF]) is updated on every index event marker and an interrupt flag (QLFG[PCE]) will be set on error that can be cleared only through software.

The index event latch configuration QEPCTL[IEL] must be configured to '00' or '11' when pcm=0 and the position counter error flag/interrupt flag are generated only in index event reset mode. The position counter value is latched into the IPOSILAT register on every index marker.

Figure 21-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or 0xF9F)



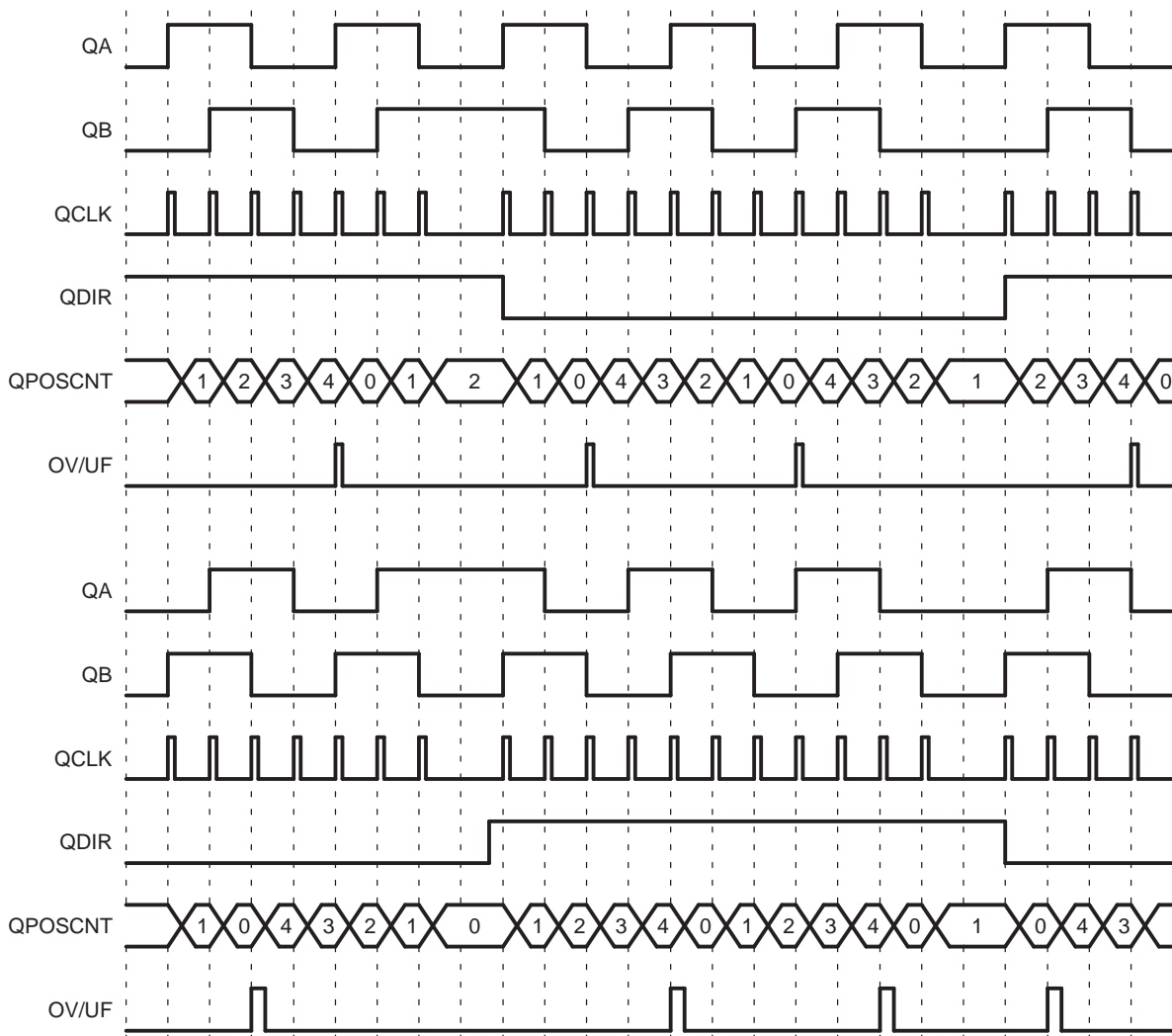
**NOTE:** In case of boundary condition where time period between Index Event and previous QCLK edge is less than SYSCLK period, then QPOSCNT gets reset to zero or QPOSMAX in the same SYSCLK cycle and does not wait for next QCLK edge to occur.

### 21.5.1.2 Position Counter Reset on Maximum Position (QEPCTL[PCRM]=01)

If the position counter is equal to QPOSMAX, then the position counter is reset to 0 on the next eQEP clock for forward movement and position counter overflow flag is set. If the position counter is equal to ZERO, then the position counter is reset to QPOSMAX on the next QEP clock for reverse movement and position-counter underflow flag is set. Figure 21-9 shows the position-counter reset operation in this mode.

The first index marker fields (QEPSTS[FIDF] and QEPSTS[FIMF]) are not applicable in this mode.

**Figure 21-9. Position Counter Underflow/Overflow (QPOSMAX = 4)**



### 21.5.1.3 Position Counter Reset on the First Index Event (QEPCTL[PCRM] = 10)

If the index event occurs during forward movement, then the position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock. Note that this is done only on the first occurrence and subsequently the position-counter value is not reset on an index event; rather, it is reset based on maximum position as described in [Section 21.5.1.2](#).

The first index marker fields (QEPSTS[FIDF] and QEPSTS[FIMF]) are not applicable in this mode.

### 21.5.1.4 Position Counter Reset on Unit Time out Event (QEPCTL[PCRM] = 11)

In this mode, QPOSCNT is set to 0 or QPOMAX, depending on the direction mode selected by QDECCTL[QSRC] bits on a unit time event. This is useful for frequency measurement.

## 21.5.2 Position Counter Latch

The eQEP index and strobe input can be configured to latch the position counter (QPOSCNT) into QPOSILAT and QPOSSLAT, respectively, on occurrence of a definite event on these pins.



### 21.5.2.1 Index Event Latch

In some applications, it may not be desirable to reset the position counter on every index event and instead it may be required to operate the position counter in full 32-bit mode (QEPCTL[PCRM] = 01 and QEPCTL[PCRM] = 10 modes).

In such cases, the eQEP position counter can be configured to latch on the following events and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker.

- Latch on Rising edge (QEPCTL[IEL]=01)
- Latch on Falling edge (QEPCTL[IEL]=10)
- Latch on Index Event Marker (QEPCTL[IEL]=11)

This is particularly useful as an error checking mechanism to check if the position counter accumulated the correct number of counts between index events. As an example, the 1000-line encoder must count 4000 times when moving in the same direction between the index events.

The index event latch interrupt flag (QFLG[IEL]) is set when the position counter is latched to the QPOSILAT register. The index event latch configuration bits (QEPCTZ[IEL]) are ignored when QEPCTL[PCRM] = 00.

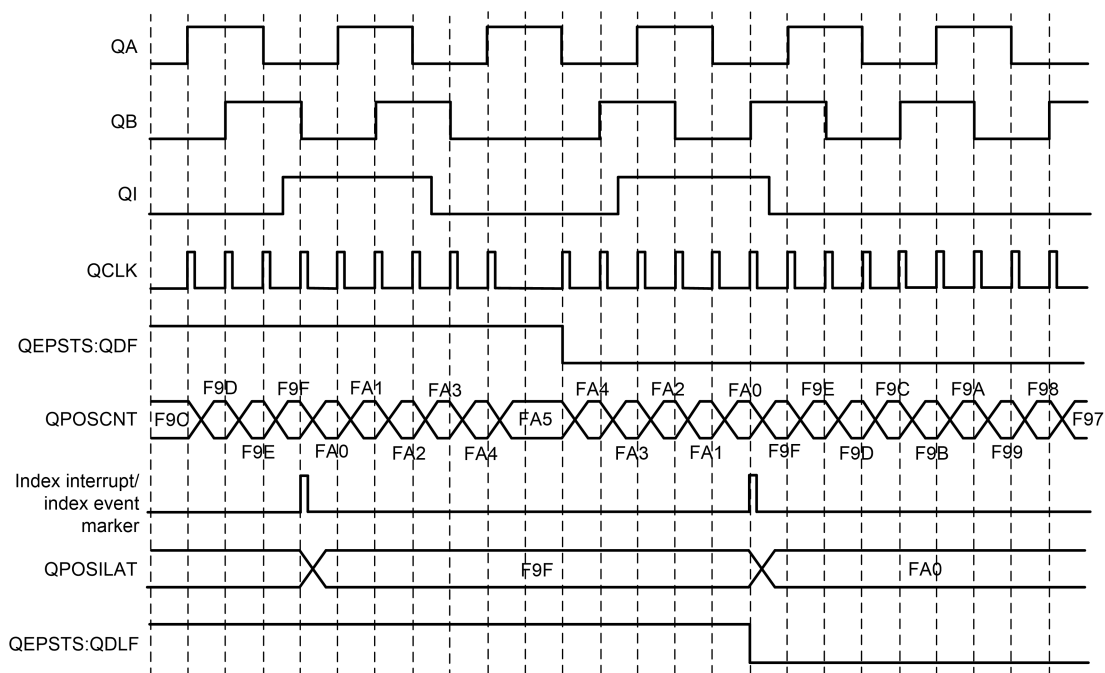
**Latch on Rising Edge (QEPCTL[IEL]=01)**— The position-counter value (QPOSCNT) is latched to the QPOSILAT register on every rising edge of an index input.

**Latch on Falling Edge (QEPCTL[IEL] = 10)**— The position-counter value (QPOSCNT) is latched to the QPOSILAT register on every falling edge of index input.

**Latch on Index Event Marker/Software Index Marker (QEPCTL[IEL] = 11)**— The first index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for latching the position counter (QEPCTL[IEL]=11).

Figure 21-10 shows the position counter latch using an index event marker.

**Figure 21-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)**



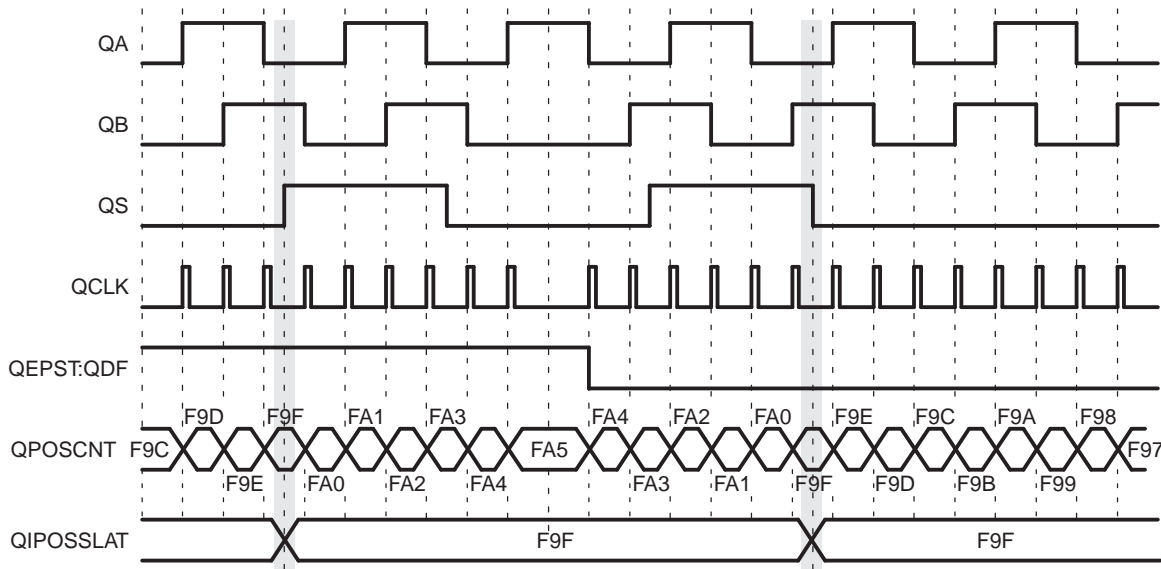
### 21.5.2.2 Strobe Event Latch

The position-counter value is latched to the QPOSSLAT register on the rising edge of the strobe input by clearing the QEPCTL[SEL] bit.

If the QEPCTL[SEL] bit is set, then the position-counter value is latched to the QPOSSLAT register on the rising edge of the strobe input for forward direction, and on the falling edge of the strobe input for reverse direction as shown in Figure 21-11.

The strobe event latch interrupt flag (QFLG[SEL]) is set when the position counter is latched to the QPOSSLAT register.

**Figure 21-11. Strobe Event Latch (QEPCTL[SEL] = 1)**



### 21.5.3 Position Counter Initialization

The position counter can be initialized using following events:

- Index event
- Strobe event
- Software initialization

**Index Event Initialization (IEI)**— The QEPI index input can be used to trigger the initialization of the position counter at the rising or falling edge of the index input. If the QEPCTL[IEI] bits are 10, then the position counter (QPOSCNT) is initialized with a value in the QPOSINIT register on the rising edge of index input. Conversely, if the QEPCTL[IEI] bits are 11, initialization will be on the falling edge of the index input.

**Strobe Event Initialization (SEI)**— If the QEPCTL[SEI] bits are 10, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input.

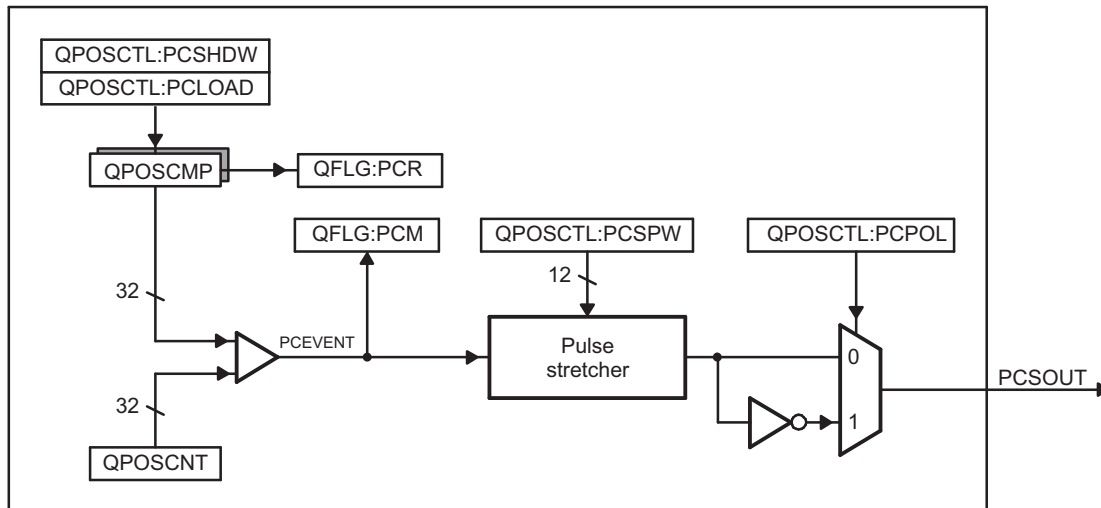
If QEPCTL[SEL] bits are 11, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

**Software Initialization (SWI)**— The position counter can be initialized in software by writing a 1 to the QEPCTL[SWI] bit. This bit is not automatically cleared. While the bit is still set, if a 1 is written to it again, the position counter will be re-initialized.

### 21.5.4 eQEP Position-compare Unit

The eQEP peripheral includes a position-compare unit that is used to generate a sync output and/or interrupt on a position-compare match. Figure 21-12 shows a diagram. The position-compare (QPOSCMP) register is shadowed and shadow mode can be enabled or disabled using the QPOSCTL[PSSHDW] bit. If the shadow mode is not enabled, the CPU writes directly to the active position compare register.

Figure 21-12. eQEP Position-compare Unit



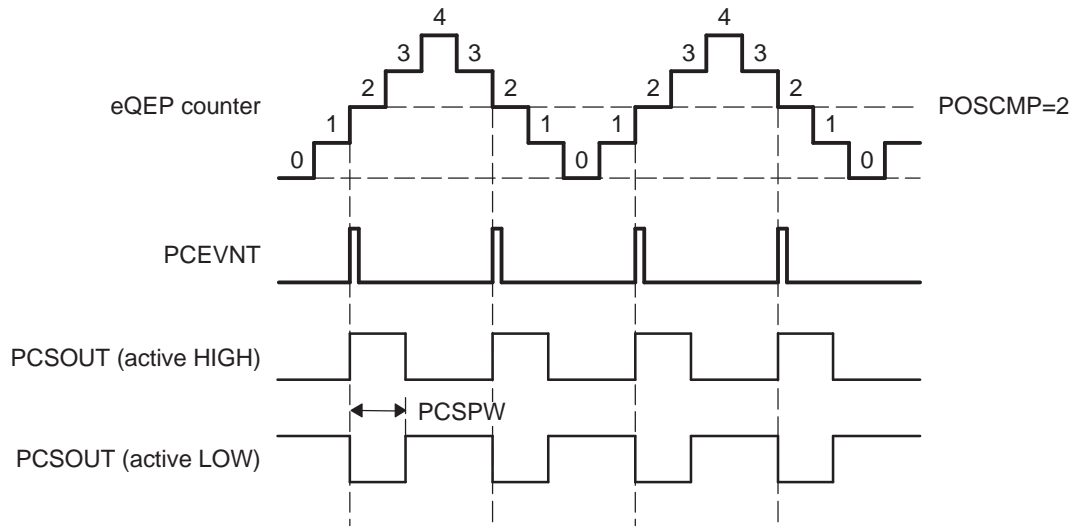
In shadow mode, you can configure the position-compare unit (QPOSCTL[PCLOAD]) to load the shadow register value into the active register on the following events, and to generate the position-compare ready (QFLG[PCR]) interrupt after loading.

- Load on compare match
- Load on position-counter zero event

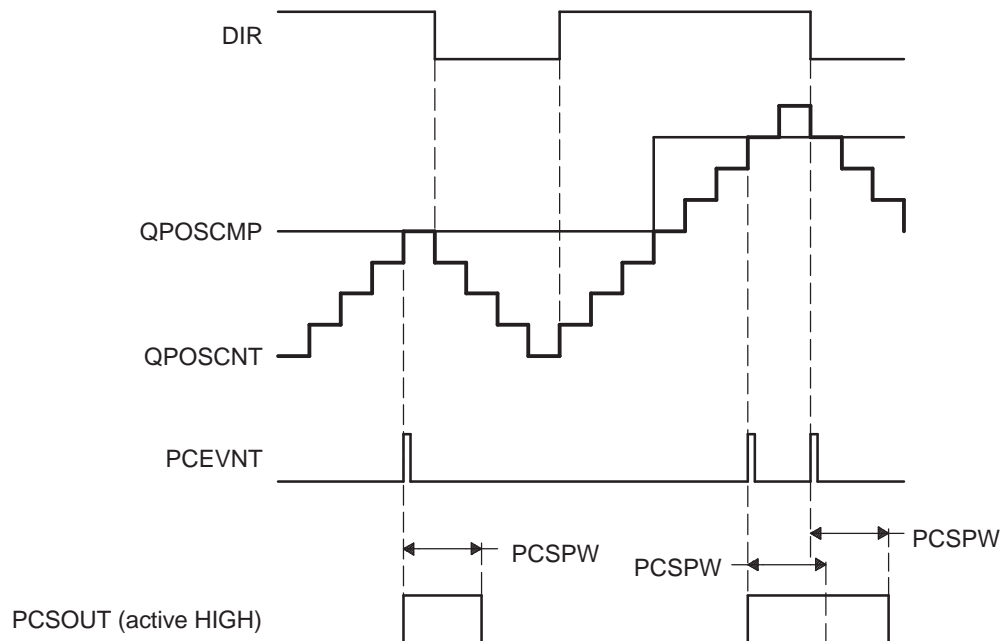
The position-compare match (QFLG[PCM]) is set when the position-counter value (QPOSCNT) matches with the active position-compare register (QPOSCMP) and the position-compare sync output of the programmable pulse width is generated on compare-match to trigger an external device.

For example, if QPOSCMP = 2, the position-compare unit generates a position-compare event on 1 to 2 transitions of the eQEP position counter for forward counting direction and on 3 to 2 transitions of the eQEP position counter for reverse counting direction (see Figure 21-13).

See the register section for the layout of the eQEP Position-Compare Control Register (QPOSCTL) and description of the QPOSCTL bit fields.

**Figure 21-13. eQEP Position-compare Event Generation Points**


The pulse stretcher logic in the position-compare unit generates a programmable position-compare sync pulse output on the position-compare match. In the event of a new position-compare match while a previous position-compare pulse is still active, then the pulse stretcher generates a pulse of specified duration from the new position-compare event as shown in [Figure 21-14](#).

**Figure 21-14. eQEP Position-compare Sync Output Pulse Stretcher**


## 21.6 eQEP Edge Capture Unit

The eQEP peripheral includes an integrated edge capture unit to measure the elapsed time between the unit position events as shown in [Figure 21-15](#). This feature is typically used for low speed measurement using the following equation:

$$v(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (6)$$

where,

- X - Unit position is defined by integer multiple of quadrature edges (see [Figure 21-16](#))
- $\Delta T$  - Elapsed time between unit position events
- v(k) - Velocity at time instant "k"

The eQEP capture timer (QCTMR) runs from prescaled SYSCLKOUT and the prescaler is programmed by the QCAPCTL[CCPS] bits. The capture timer (QCTMR) value is latched into the capture period register (QCPRD) on every unit position event and then the capture timer is reset, a flag is set in QEPSTS:UPEVNT to indicate that new value is latched into the QCPRD register. Software can check this status flag before reading the period register for low speed measurement, and clear the flag by writing 1.

Time measurement ( $\Delta T$ ) between unit position events will be correct if the following conditions are met:

- No more than 65,535 counts have occurred between unit position events.
- No direction change between unit position events.

The capture unit sets the eQEP overflow error flag (QEPSTS[COEF]) in the event of capture timer overflow between unit position events. If a direction change occurs between the unit position events, then an error flag is set in the status register (QEPSTS[CDEF]).

The Capture Timer (QCTMR) and Capture Period register (QCPRD) can be configured to latch on following events.

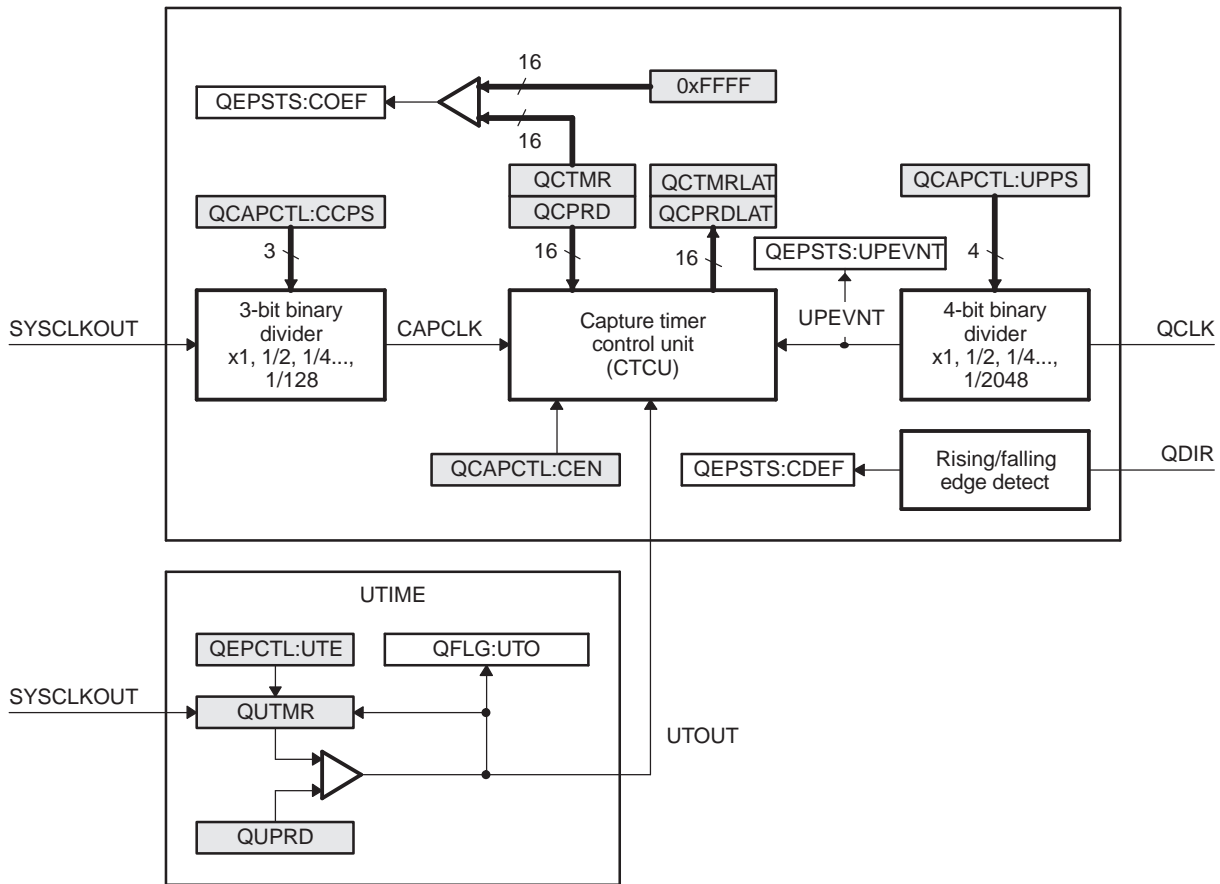
- CPU read of QPOSCNT register
- Unit time-out event

If the QEPCTL[QCLM] bit is cleared, then the capture timer and capture period values are latched into the QCTMRLAT and QCPRDLAT registers, respectively, when the CPU reads the position counter (QPOSCNT).

If the QEPCTL[QCLM] bit is set, then the position counter, capture timer, and capture period values are latched into the QPOSLAT, QCTMRLAT and QCPRDLAT registers, respectively, on unit time out.

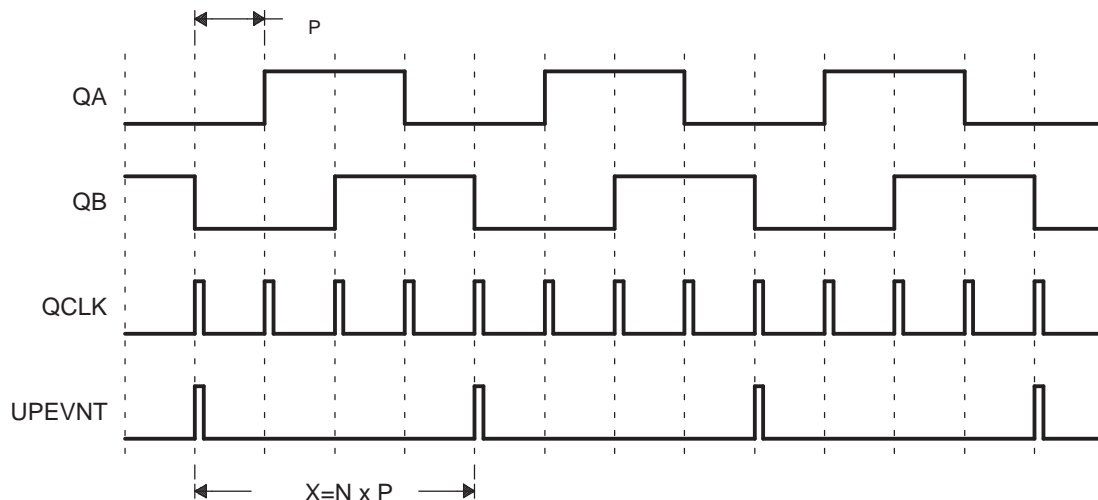
[Figure 21-17](#) shows the capture unit operation along with the position counter.

Figure 21-15. eQEP Edge Capture Unit



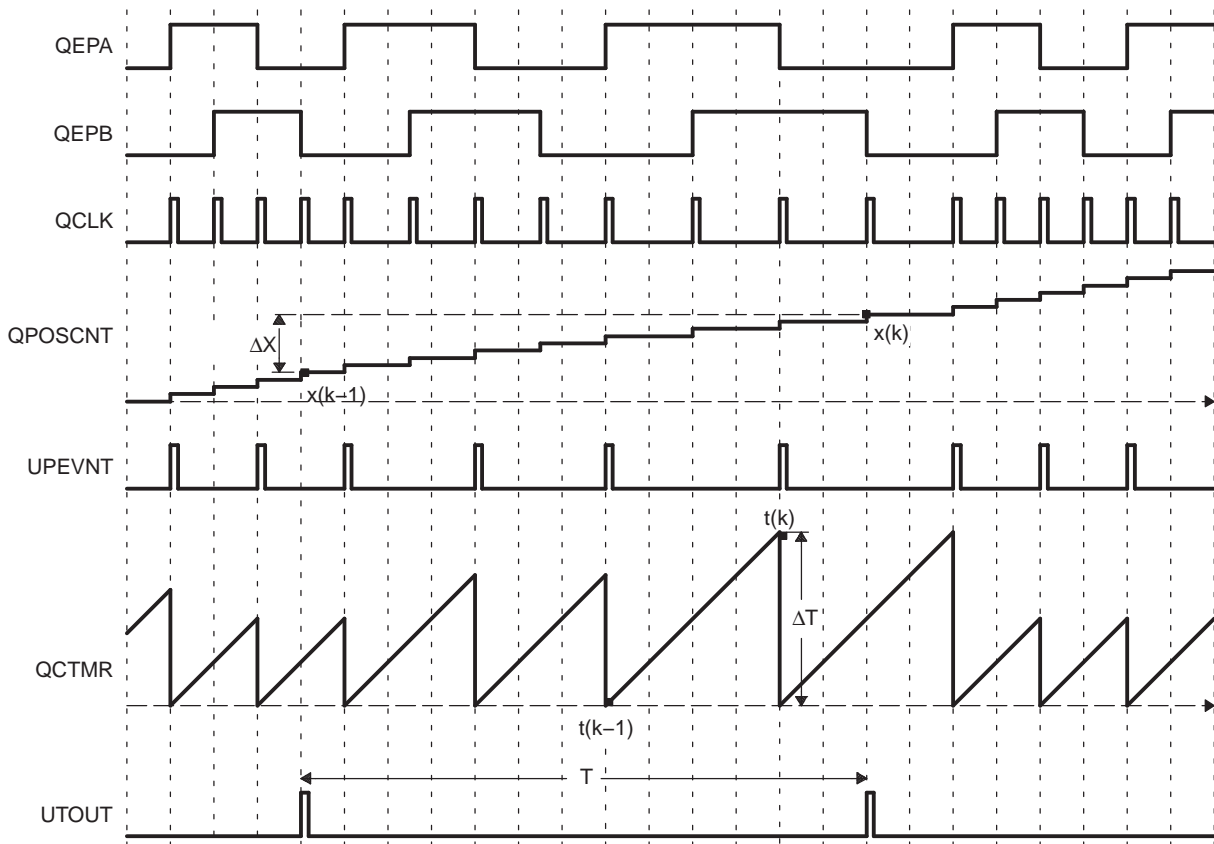
**NOTE:** The QCAPCTL[UPPS] prescaler should not be modified dynamically (such as switching the unit event prescaler from QCLK/4 to QCLK/8). Doing so may result in undefined behavior. The QCAPCTL[CPPS] prescaler can be modified dynamically (such as switching CAPCLK prescaling mode from SYSCLK/4 to SYSCLK/8) only after the capture unit is disabled.

Figure 21-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010)



A N - Number of quadrature periods selected using QCAPCTL[UPPS] bits

Figure 21-17. eQEP Edge Capture Unit - Timing Details



Velocity calculation equations:

$$v(k) = \frac{x(k) - x(k - 1)}{T} = \frac{\Delta X}{T} \quad (7)$$

where

$v(k)$ : Velocity at time instant  $k$

$x(k)$ : Position at time instant  $k$

$x(k-1)$ : Position at time instant  $k-1$

$T$ : Fixed unit time or inverse of velocity calculation rate

$\Delta X$ : Incremental position movement in unit time

$X$ : Fixed unit position

$\Delta T$ : Incremental time elapsed for unit position movement

$t(k)$ : Time instant "k"

$t(k-1)$ : Time instant "k-1"

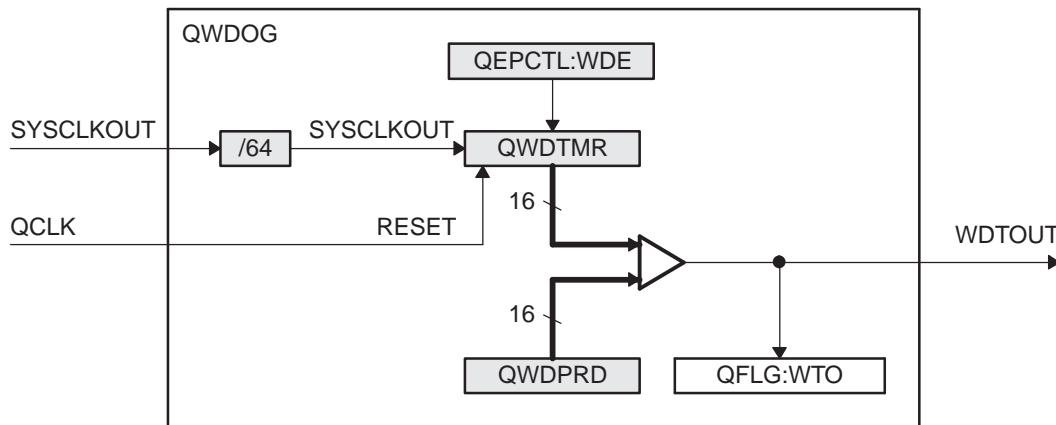
Unit time ( $T$ ) and unit period( $X$ ) are configured using the QUPRD and QCAPCTL[UPPS] registers. Incremental position output and incremental time output is available in the QPOSLAT and QCPRDLAT registers.

Parameter	Relevant Register to Configure or Read the Information
T	Unit Period Register (QUPRD)
$\Delta X$	Incremental Position = QPOSLAT(k) - QPOSLAT(K-1)
X	Fixed unit position defined by sensor resolution and ZCAPCTL[Upps] bits
$\Delta T$	Capture Period Latch (QCPRDLAT)

## 21.7 eQEP Watchdog

The eQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature-clock to indicate proper operation of the motion-control system. The eQEP watchdog timer is clocked from SYSCLKOUT/64 and the quadrature clock event (pulse) resets the watchdog timer. If no quadrature-clock event is detected until a period match ( $QWDPRD = QWDTMR$ ), then the watchdog timer will time out and the watchdog interrupt flag will be set (QFLG[WTO]). The time-out value is programmable through the watchdog period register (QWDPRD).

**Figure 21-18. eQEP Watchdog Timer**



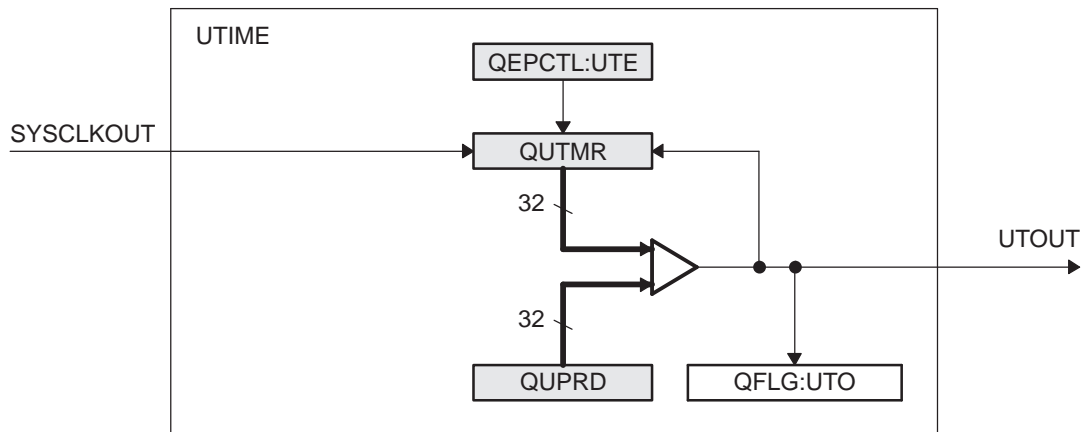
## 21.8 Unit Timer Base

The eQEP peripheral includes a 32-bit timer (QUTMR) that is clocked by SYSCLKOUT to generate periodic interrupts for velocity calculations. Whenever the unit timer (QUTMR) matches the unit period register (QUPRD), it resets the unit timer (QUTMR) and also generates the unit time out interrupt flag (QFLG[UTO]). The unit timer gets reset whenever timer value equals to configured period value.

The eQEP peripheral can be configured to latch the position counter, capture timer, and capture period values on a unit time out event so that latched values are used for velocity calculation as described in [Section 21.6](#).



Figure 21-19. eQEP Unit Time Base



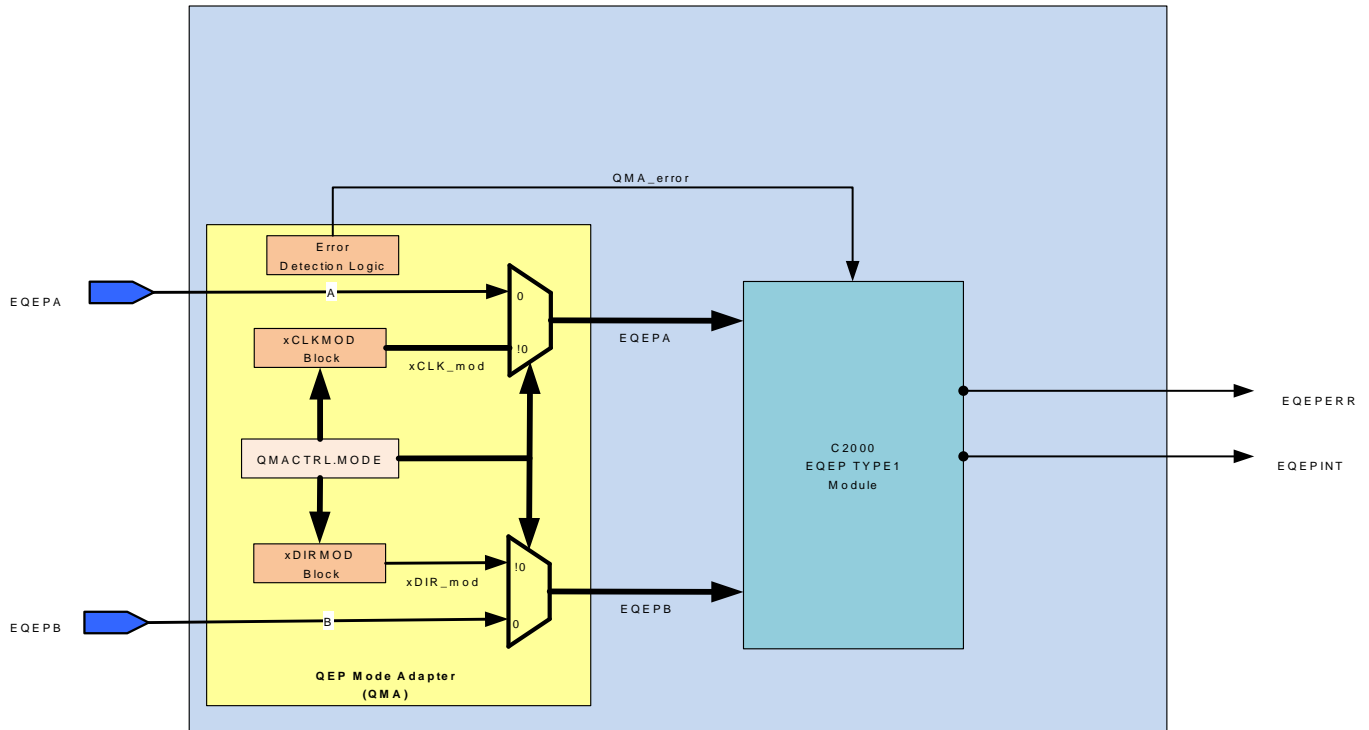
## 21.9 QMA Module

The QEP Mode Adapter (QMA) is designed to extend the C2000 eQEP module capabilities to support the additional modes described below. [Figure 21-20](#) depicts how the QMA module is integrated into the C2000 eQEP module.

At reset, by default QMA logic is bypassed and the EQEPA and EQEPB inputs from the pins go directly into the eQEP module. When QMA module is enabled by configuring the QMACTRL[MODE] register, the EQEPA and EQEPB input are processed by this module and modified version of EQEPA and EQEPB signals are sent to the eQEP module. The QMA module requires the eQEP module to be configured in the Direction-Count mode and generates a clock signal on EQEPA input and direction signal on EQEPB input as needed for the proper operation of the intended mode.

- The xCLKMOD block inside the QMA module looks at the transitions on external EQEPA and EQEPB signals to generate the clock signal on the EQEPA input to the eQEP module.
- The xDIRMOD block inside the QMA module looks at the transitions on external EQEPA and EQEPB signals to generate the direction signal on the EQEPB input to the eQEP module.

The QMA module has error detection logic to detect illegal transitions on EQEPA and EQEPB input signals. The QMA module's error and interrupt are integrated inside the eQEP module as described in the eQEP Interrupt Structure section. In addition QMACTRL register configuration can be locked using QMALOCK register. Refer to the register description for more details.

**Figure 21-20. QMA Module Block Diagram**


### 21.9.1 Modes of Operation

The QMA module can be operated in the following modes by configuring the QMACTRL register.

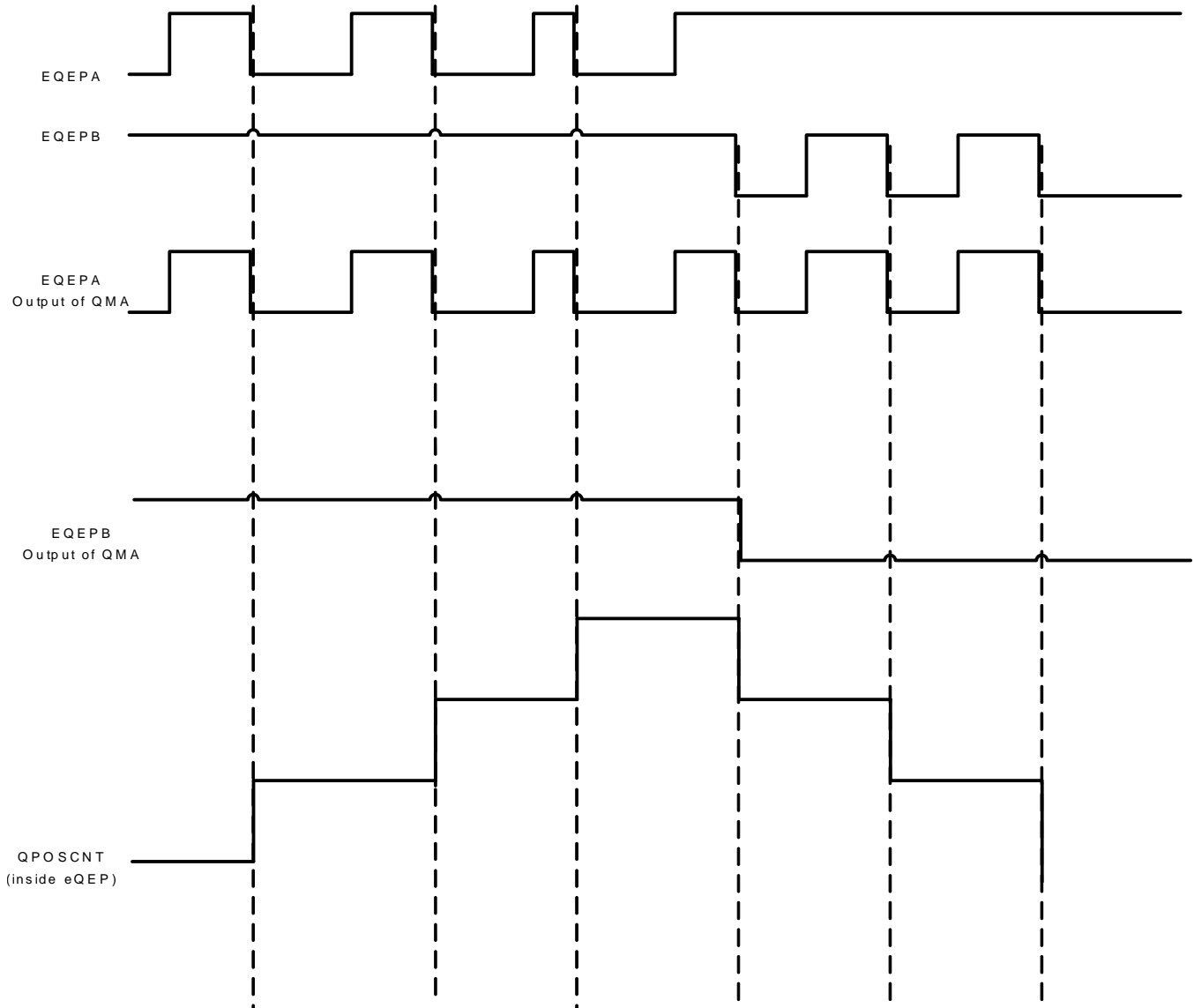
#### 21.9.1.1 QMA Mode-1(QMACTRL[MODE]=1)

In this mode outputs of QMA correspond to the following as shown in [Figure 21-21](#):

- EQEPA Output of QMA is AND of EQEPA and EQEPB inputs coming from the pin
- EQEPB Output of QMA is direction signal generated by QMA based on EQEPA and EQEPB inputs

This mode is used when the default state of EQEPA and EQEPB inputs is high

**Figure 21-21. QMA Mode-1**

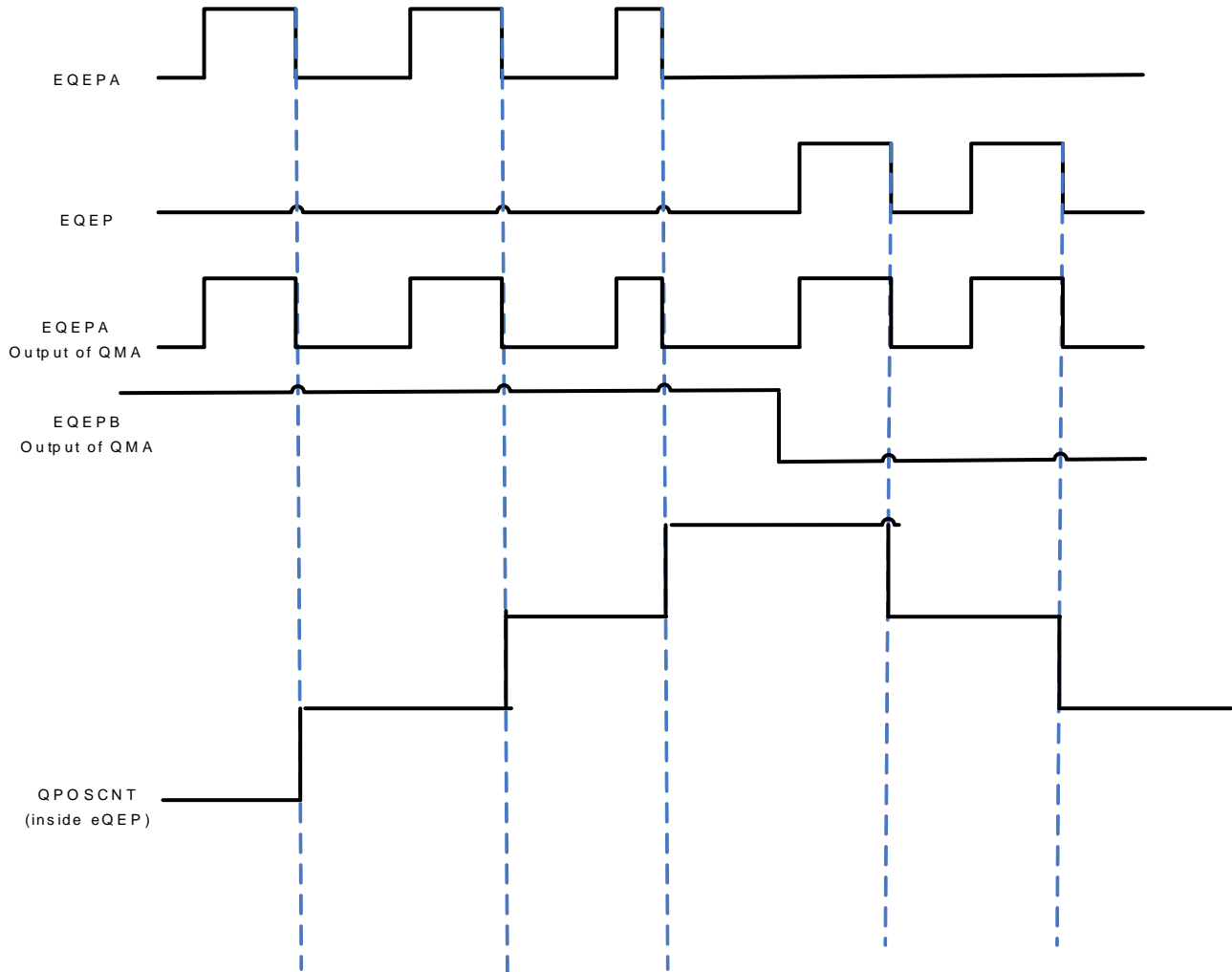


**21.9.1.2 QMA Mode-2(QMACTRL[MODE]=2)**

In this mode, outputs of QMA correspond to the following as shown in [Figure 21-22](#):

- EQEPA Output of QMA is OR of EQEPA and EQEPB inputs coming from the pin
- EQEPB Output of QMA is direction signal generated by QMA based on EQEPA and EQEPB inputs

This mode is used when the default state of EQEPA and EQEPB inputs is low

**Figure 21-22. QMA Mode-2**


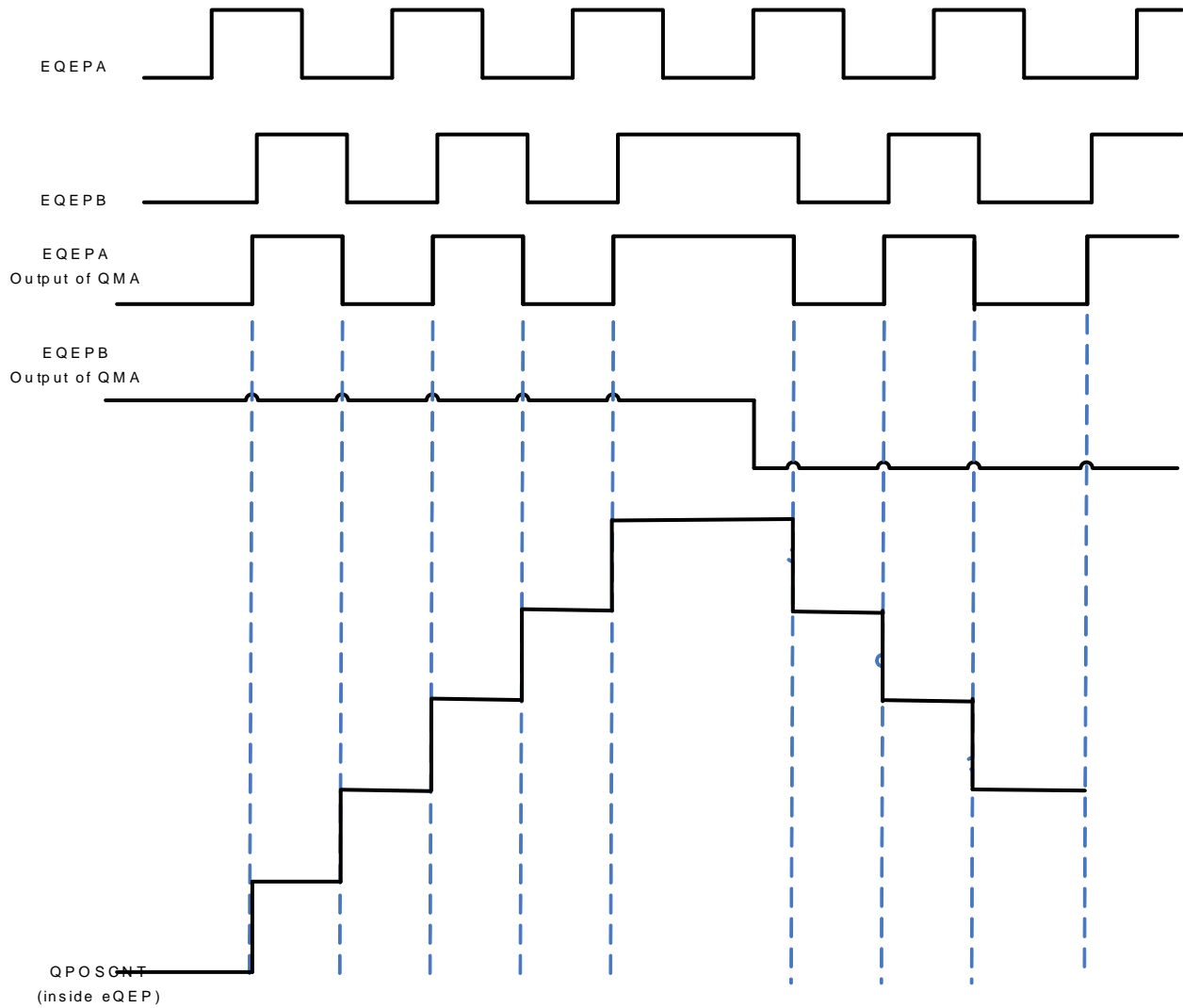
### 21.9.1.3 QMA Mode-3(QMACTRL[MODE]=3)

In this mode outputs of QMA correspond to the following as shown in [Figure 21-23](#)

- EQEPA Output of QMA is the same as EQEPB input signal coming from the pin
- EQEPB Output of QMA is direction signal generated by QMA based on EQEPA and EQEPB inputs

This mode is used when the default state of EQEPA and EQEPB inputs are in quadrature but EQEPB input alone is used for pulse counting.

Figure 21-23. QMA Mode-3



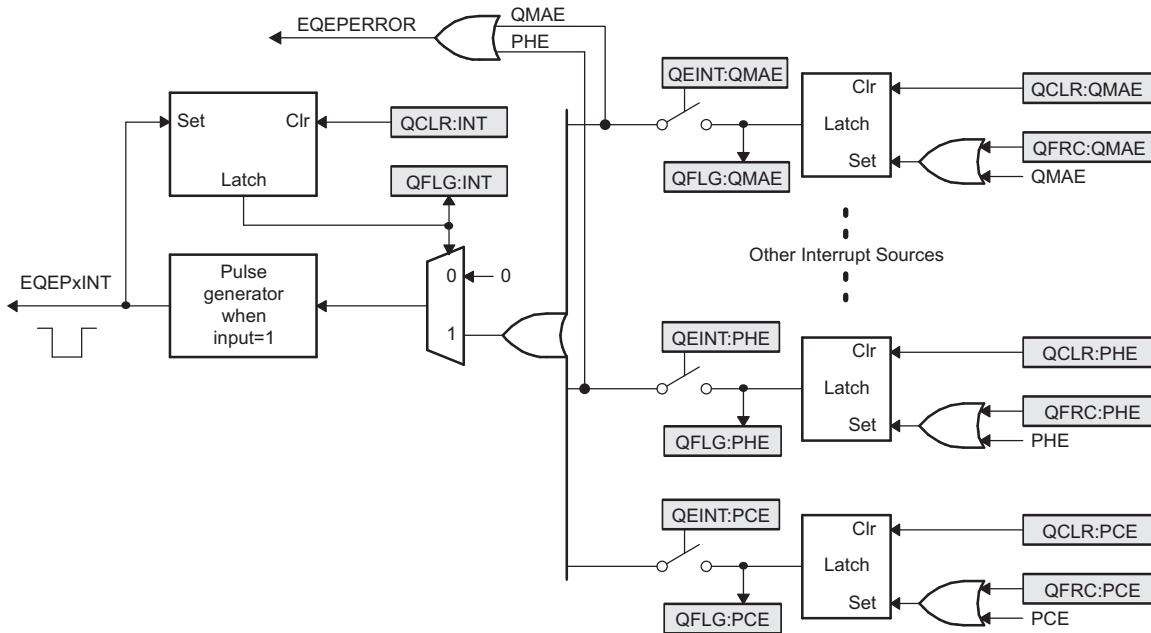
### 21.9.2 Interrupt and Error Generation

The error detection logic detects illegal transitions on EQEPA and EQEPB signals and generates an error signal. This error signal can be used to generate eQEP interrupt and error output. Refer to [Section 21.10](#) for details.

### 21.10 eQEP Interrupt Structure

[Figure 21-24](#) shows how the interrupt mechanism works in the EQEP module.

Figure 21-24. EQEP Interrupt Generation



Eleven interrupt events (PCE, PHE, QDC, WTO, PCU, PCO, PCR, PCM, SEL, IEL and UTO) can be generated. The interrupt control register (QEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (QFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT).

An Interrupt pulse is generated to PIE when:

- a. Interrupt is enabled for eQEP event inside QEINT register
- b. Interrupt flag for eQEP event inside QFLG register is set, and
- c. Global interrupt status flag bit QFLG[INT] had been cleared for previously generated interrupt event. The interrupt service routine will need to clear the global interrupt flag bit and the serviced event, via the interrupt clear register (QCLR), before any other interrupt pulses are generated. If either flags inside the QFLG register are not cleared, further interrupt event will not generate interrupt to PIE. You can force an interrupt event by way of the interrupt force register (QFRC), which is useful for test purposes.

## 21.11 eQEP Registers

This section describes the Enhanced Quadrature Encoder Pulse Registers.

### 21.11.1 eQEP Base Addresses

**Table 21-3. eQEP Base Address Table**

Device Registers	Register Name	Start Address	End Address
EQep1Regs	EQEP_REGS	0x0000_5100	0x0000_513F
EQep2Regs	EQEP_REGS	0x0000_5140	0x0000_517F

### 21.11.2 EQEP\_REGS Registers

Table 21-4 lists the EQEP\_REGS registers. All register offset addresses not listed in Table 21-4 should be considered as reserved locations and the register contents should not be modified.

**Table 21-4. EQEP\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	QPOSCNT	Position Counter		<a href="#">Go</a>
2h	QPOSINIT	Position Counter Init		<a href="#">Go</a>
4h	QPOSMAX	Maximum Position Count		<a href="#">Go</a>
6h	QPOSCMP	Position Compare		<a href="#">Go</a>
8h	QPOSILAT	Index Position Latch		<a href="#">Go</a>
Ah	QPOSSLAT	Strobe Position Latch		<a href="#">Go</a>
Ch	QPOSLAT	Position Latch		<a href="#">Go</a>
Eh	QUTMR	QEP Unit Timer		<a href="#">Go</a>
10h	QUPRD	QEP Unit Period		<a href="#">Go</a>
12h	QWDTMR	QEP Watchdog Timer		<a href="#">Go</a>
13h	QWDPRD	QEP Watchdog Period		<a href="#">Go</a>
14h	QDECCTL	Quadrature Decoder Control		<a href="#">Go</a>
15h	QEPCTL	QEP Control		<a href="#">Go</a>
16h	QCAPCTL	Quadrature Capture Control		<a href="#">Go</a>
17h	QPOSCTL	Position Compare Control		<a href="#">Go</a>
18h	QEINT	QEP Interrupt Control		<a href="#">Go</a>
19h	QFLG	QEP Interrupt Flag		<a href="#">Go</a>
1Ah	QCLR	QEP Interrupt Clear		<a href="#">Go</a>
1Bh	QFRC	QEP Interrupt Force		<a href="#">Go</a>
1Ch	QEPSTS	QEP Status		<a href="#">Go</a>
1Dh	QCTMR	QEP Capture Timer		<a href="#">Go</a>
1Eh	QCPRD	QEP Capture Period		<a href="#">Go</a>
1Fh	QCTMLAT	QEP Capture Latch		<a href="#">Go</a>
20h	QCPRDLAT	QEP Capture Period Latch		<a href="#">Go</a>
30h	REV	QEP Revision Number		<a href="#">Go</a>
32h	QEPSTROBESEL	QEP Strobe select register		<a href="#">Go</a>
34h	QMACTRL	QMA Control register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 21-5 shows the codes that are used for access types in this section.

**Table 21-5. EQEP\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		



**Table 21-5. EQEP\_REGS Access Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**21.11.2.1 QPOSCNT Register (Offset = 0h) [reset = 0h]**

QPOSCNT is shown in [Figure 21-25](#) and described in [Table 21-6](#).

Return to the [Summary Table](#).

Position Counter

**Figure 21-25. QPOSCNT 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
QPOSCNT																															
R/W-0h																															

**Table 21-6. QPOSCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSCNT	R/W	0h	<p>Position Counter</p> <p>This 32-bit position counter register counts up/down on every eQEP pulse based on direction input. This counter acts as a position integrator whose count value is proportional to position from a give reference point. This Register acts as a Read ONLY register while counter is counting up/down.</p> <p>Note: It is recommended to only write to the position counter register (QPOSCNT) during initialization, i.e. when the eQEP position counter is disabled (QPEN bit of QEPCTL is zero). Once the position counter is enabled (QPEN bit is one), writing to the eQEP position counter register (QPOSCNT) may cause unexpected results.</p> <p>Reset type: SYSRSn</p>

### 21.11.2.2 QPOSINIT Register (Offset = 2h) [reset = 0h]

QPOSINIT is shown in [Figure 21-26](#) and described in [Table 21-7](#).

Return to the [Summary Table](#).

Position Counter Init

**Figure 21-26. QPOSINIT 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
QPOSINIT																															
R/W-0h																															

**Table 21-7. QPOSINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSINIT	R/W	0h	Position Counter Init This register contains the position value that is used to initialize the position counter based on external strobe or index event. The position counter can be initialized through software. Writes to this register should always be full 32-bit writes. Reset type: SYSRSn

### 21.11.2.3 QPOSMAX Register (Offset = 4h) [reset = 0h]

QPOSMAX is shown in [Figure 21-27](#) and described in [Table 21-8](#).

Return to the [Summary Table](#).

Maximum Position Count

**Figure 21-27. QPOSMAX 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
QPOSMAX																															
R/W-0h																															

**Table 21-8. QPOSMAX Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSMAX	R/W	0h	Maximum Position Count This register contains the maximum position counter value. Writes to this register should always be full 32-bit writes. Reset type: SYSRSn

**21.11.2.4 QPOSCMP Register (Offset = 6h) [reset = 0h]**

QPOSCMP is shown in [Figure 21-28](#) and described in [Table 21-9](#).

Return to the [Summary Table](#).

Position Compare

**Figure 21-28. QPOSCMP 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
QPOSCMP																															
R/W-0h																															

**Table 21-9. QPOSCMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSCMP	R/W	0h	Position Compare The position-compare value in this register is compared with the position counter (QPOSCNT) to generate sync output and/or interrupt on compare match. Reset type: SYSRSn

### 21.11.2.5 QPOSILAT Register (Offset = 8h) [reset = 0h]

QPOSILAT is shown in [Figure 21-29](#) and described in [Table 21-10](#).

Return to the [Summary Table](#).

Index Position Latch

**Figure 21-29. QPOSILAT 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
QPOSILAT																															
R-0h																															

**Table 21-10. QPOSILAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSILAT	R	0h	Index Position Latch The position-counter value is latched into this register on an index event as defined by the QEPCTL[IEL] bits. Reset type: SYSRSn

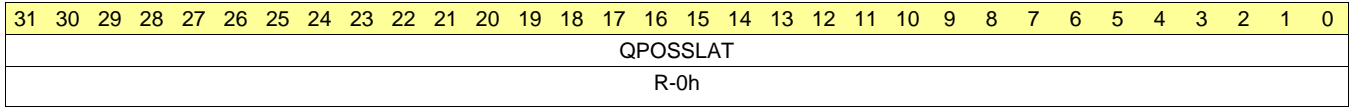
**21.11.2.6 QPOSSLAT Register (Offset = Ah) [reset = 0h]**

QPOSSLAT is shown in [Figure 21-30](#) and described in [Table 21-11](#).

Return to the [Summary Table](#).

Strobe Position Latch

**Figure 21-30. QPOSSLAT Register**



**Table 21-11. QPOSSLAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSSLAT	R	0h	Strobe Position Latch The position-counter value is latched into this register on a strobe event as defined by the QEPCTL[SEL] bits. Reset type: SYSRSn

**21.11.2.7 QPOSLAT Register (Offset = Ch) [reset = 0h]**

QPOSLAT is shown in [Figure 21-31](#) and described in [Table 21-12](#).

Return to the [Summary Table](#).

Position Latch

**Figure 21-31. QPOSLAT 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
QPOSLAT																															
R-0h																															

**Table 21-12. QPOSLAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSLAT	R	0h	Position Latch The position-counter value is latched into this register on a unit time out event. Reset type: SYSRSn



**21.11.2.8 QUTMR Register (Offset = Eh) [reset = 0h]**

QUTMR is shown in [Figure 21-32](#) and described in [Table 21-13](#).

Return to the [Summary Table](#).

QEP Unit Timer

**Figure 21-32. QUTMR 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
QUTMR																															
R/W-0h																															

**Table 21-13. QUTMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QUTMR	R/W	0h	<p>QEP Unit Timer</p> <p>This register acts as time base for unit time event generation. When this timer value matches the unit time period value a unit time event is generated.</p> <p>Reset type: SYSRSn</p>

### 21.11.2.9 QUPRD Register (Offset = 10h) [reset = 0h]

QUPRD is shown in [Figure 21-33](#) and described in [Table 21-14](#).

Return to the [Summary Table](#).

QEP Unit Period

**Figure 21-33. QUPRD 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
QUPRD																															
R/W-0h																															

**Table 21-14. QUPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QUPRD	R/W	0h	QEP Unit Period This register contains the period count for the unit timer to generate periodic unit time events. These events latch the eQEP position information at periodic intervals and optionally generate an interrupt. Writes to this register should always be full 32-bit writes. Reset type: SYSRSn

**21.11.2.10 QWDTMR Register (Offset = 12h) [reset = 0h]**

QWDTMR is shown in [Figure 21-34](#) and described in [Table 21-15](#).

Return to the [Summary Table](#).

QEP Watchdog Timer

**Figure 21-34. QWDTMR Register**

15	14	13	12	11	10	9	8
QWDTMR							
R/W-0h							
7	6	5	4	3	2	1	0
QWDTMR							
R/W-0h							

**Table 21-15. QWDTMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QWDTMR	R/W	0h	<p>QEP Watchdog Timer</p> <p>This register acts as time base for the watchdog to detect motor stalls. When this timer value matches with the watchdog's period value a watchdog timeout interrupt is generated. This register is reset upon edge transition in quadrature-clock indicating the motion.</p> <p>Reset type: SYSRSn</p>

**21.11.2.11 QWDPRD Register (Offset = 13h) [reset = 0h]**

QWDPRD is shown in [Figure 21-35](#) and described in [Table 21-16](#).

Return to the [Summary Table](#).

QEP Watchdog Period

**Figure 21-35. QWDPRD Register**

15	14	13	12	11	10	9	8
QWDPRD							
R/W-0h							
7	6	5	4	3	2	1	0
QWDPRD							
R/W-0h							

**Table 21-16. QWDPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QWDPRD	R/W	0h	QEP Watchdog Period This register contains the time-out count for the eQEP peripheral watch dog timer. When the watchdog timer value matches the watchdog period value, a watchdog timeout interrupt is generated. Reset type: SYSRSn

### 21.11.2.12 QDECCTL Register (Offset = 14h) [reset = 0h]

QDECCTL is shown in [Figure 21-36](#) and described in [Table 21-17](#).

Return to the [Summary Table](#).

Quadrature Decoder Control

**Figure 21-36. QDECCTL Register**

15	14	13	12	11	10	9	8
QSRC		SOEN	SPSEL	XCR	SWAP	IGATE	QAP
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
QBP	QIP	QSP	RESERVED				
R/W-0h	R/W-0h	R/W-0h	R-0h				

**Table 21-17. QDECCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	QSRC	R/W	0h	Position-counter source selection Reset type: SYSRSn 0h (R/W) = Quadrature count mode (QCLK = iCLK, QDIR = iDIR) 1h (R/W) = Direction-count mode (QCLK = xCLK, QDIR = xDIR) 2h (R/W) = UP count mode for frequency measurement (QCLK = xCLK, QDIR = 1) 3h (R/W) = DOWN count mode for frequency measurement (QCLK = xCLK, QDIR = 0)
13	SOEN	R/W	0h	Sync output-enable Reset type: SYSRSn 0h (R/W) = Disable position-compare sync output 1h (R/W) = Enable position-compare sync output
12	SPSEL	R/W	0h	Sync output pin selection Reset type: SYSRSn 0h (R/W) = Index pin is used for sync output 1h (R/W) = Strobe pin is used for sync output
11	XCR	R/W	0h	External Clock Rate Reset type: SYSRSn 0h (R/W) = 2x resolution: Count the rising/falling edge 1h (R/W) = 1x resolution: Count the rising edge only
10	SWAP	R/W	0h	CLK/DIR Signal Source for Position Counter Reset type: SYSRSn 0h (R/W) = Quadrature-clock inputs are not swapped 1h (R/W) = Quadrature-clock inputs are swapped
9	IGATE	R/W	0h	Index pulse gating option Reset type: SYSRSn 0h (R/W) = Disable gating of Index pulse 1h (R/W) = Gate the index pin with strobe
8	QAP	R/W	0h	QEPA input polarity Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Negates QEPA input
7	QBP	R/W	0h	QEPB input polarity Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Negates QEPB input

**Table 21-17. QDECCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	QIP	R/W	0h	QEPI input polarity Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Negates QEPI input
5	QSP	R/W	0h	QEPS input polarity Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Negates QEPS input
4-0	RESERVED	R	0h	Reserved

**21.11.2.13 QEPCTL Register (Offset = 15h) [reset = 0h]**

 QEPCTL is shown in [Figure 21-37](#) and described in [Table 21-18](#).

 Return to the [Summary Table](#).

QEP Control

**Figure 21-37. QEPCTL Register**

15		14		13		12		11		10		9		8	
FREE_SOFT				PCRM				SEI				IEI			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			
7		6		5		4		3		2		1		0	
SWI		SEL		IEL		QPEN		QCLM		UTE		WDE			
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h			

**Table 21-18. QEPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation mode Reset type: SYSRSn 0h (R/W) = QPOSCNT behavior Position counter stops immediately on emulation suspend 0h (R/W) = QWDTMR behavior Watchdog counter stops immediately 0h (R/W) = QUTMR behavior Unit timer stops immediately 0h (R/W) = QCTMR behavior Capture Timer stops immediately 1h (R/W) = QPOSCNT behavior Position counter continues to count until the rollover 1h (R/W) = QWDTMR behavior Watchdog counter counts until WD period match roll over 1h (R/W) = QUTMR behavior Unit timer counts until period rollover 1h (R/W) = QCTMR behavior Capture Timer counts until next unit period event 2h (R/W) = QPOSCNT behavior Position counter is unaffected by emulation suspend 2h (R/W) = QWDTMR behavior Watchdog counter is unaffected by emulation suspend 2h (R/W) = QUTMR behavior Unit timer is unaffected by emulation suspend 2h (R/W) = QCTMR behavior Capture Timer is unaffected by emulation suspend 3h (R/W) = Same as FREE_SOFT_2
13-12	PCRM	R/W	0h	Position counter reset Reset type: SYSRSn 0h (R/W) = Position counter reset on an index event 1h (R/W) = Position counter reset on the maximum position 2h (R/W) = Position counter reset on the first index event 3h (R/W) = Position counter reset on a unit time event
11-10	SEI	R/W	0h	Strobe event initialization of position counter Reset type: SYSRSn 0h (R/W) = Does nothing (action disabled) 1h (R/W) = Does nothing (action disabled) 2h (R/W) = Initializes the position counter on rising edge of the QEPS signal 3h (R/W) = Clockwise Direction: Initializes the position counter on the rising edge of QEPS strobe Counter Clockwise Direction: Initializes the position counter on the falling edge of QEPS strobe

**Table 21-18. QEPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	IEI	R/W	0h	Index event init of position count Reset type: SYSRSn 0h (R/W) = Do nothing (action disabled) 1h (R/W) = Do nothing (action disabled) 2h (R/W) = Initializes the position counter on the rising edge of the QEPI signal (QPOSCNT = QPOSINIT) 3h (R/W) = Initializes the position counter on the falling edge of QEPI signal (QPOSCNT = QPOSINIT)
7	SWI	R/W	0h	Software init position counter Reset type: SYSRSn 0h (R/W) = Do nothing (action disabled) 1h (R/W) = Initialize position counter (QPOSCNT=QPOSINIT). This bit is not cleared automatically
6	SEL	R/W	0h	Strobe event latch of position counter Reset type: SYSRSn 0h (R/W) = The position counter is latched on the rising edge of QEPS strobe (QPOSSLAT = POSCNT). Latching on the falling edge can be done by inverting the strobe input using the QSP bit in the QDECCTL register 1h (R/W) = Clockwise Direction: Position counter is latched on rising edge of QEPS strobe Counter Clockwise Direction: Position counter is latched on falling edge of QEPS strobe
5-4	IEL	R/W	0h	Index event latch of position counter (software index marker) Reset type: SYSRSn 0h (R/W) = Reserved 1h (R/W) = Latches position counter on rising edge of the index signal 2h (R/W) = Latches position counter on falling edge of the index signal 3h (R/W) = Software index marker. Latches the position counter and quadrature direction flag on index event marker. The position counter is latched to the QPOSILAT register and the direction flag is latched in the QEPSTS[QDLF] bit. This mode is useful for software index marking.
3	QPEN	R/W	0h	Quadrature position counter enable/software reset Reset type: SYSRSn 0h (R/W) = Reset the eQEP peripheral internal operating flags/read-only registers. Control/configuration registers are not disturbed by a software reset. When QPEN is disabled, some flags in the QFLG register do not get reset or cleared and show the actual state of that flag. 1h (R/W) = eQEP position counter is enabled
2	QCLM	R/W	0h	QEP capture latch mode Reset type: SYSRSn 0h (R/W) = Latch on position counter read by CPU. Capture timer and capture period values are latched into QCTMRLAT and QCPRDLAT registers when CPU reads the QPOSCNT register. 1h (R/W) = Latch on unit time out. Position counter, capture timer and capture period values are latched into QPOSILAT, QCTMRLAT and QCPRDLAT registers on unit time out.
1	UTE	R/W	0h	QEP unit timer enable Reset type: SYSRSn 0h (R/W) = Disable eQEP unit timer 1h (R/W) = Enable unit timer



**Table 21-18. QEPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	WDE	R/W	0h	QEP watchdog enable Reset type: SYSRSn 0h (R/W) = Disable the eQEP watchdog timer 1h (R/W) = Enable the eQEP watchdog timer

**21.11.2.14 QCAPCTL Register (Offset = 16h) [reset = 0h]**

QCAPCTL is shown in [Figure 21-38](#) and described in [Table 21-19](#).

Return to the [Summary Table](#).

Quadrature Capture Control

**Figure 21-38. QCAPCTL Register**

15	14	13	12	11	10	9	8
CEN	RESERVED						
R/W-0h	R-0h						
7	6	5	4	3	2	1	0
RESERVED	CCPS			UPPS			
R-0h	R/W-0h			R/W-0h			

**Table 21-19. QCAPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	CEN	R/W	0h	Enable eQEP capture Reset type: SYSRSn 0h (R/W) = eQEP capture unit is disabled 1h (R/W) = eQEP capture unit is enabled
14-7	RESERVED	R	0h	Reserved
6-4	CCPS	R/W	0h	eQEP capture timer clock prescaler Reset type: SYSRSn 0h (R/W) = CAPCLK = SYSCLKOUT/1 1h (R/W) = CAPCLK = SYSCLKOUT/2 2h (R/W) = CAPCLK = SYSCLKOUT/4 3h (R/W) = CAPCLK = SYSCLKOUT/8 4h (R/W) = CAPCLK = SYSCLKOUT/16 5h (R/W) = CAPCLK = SYSCLKOUT/32 6h (R/W) = CAPCLK = SYSCLKOUT/64 7h (R/W) = CAPCLK = SYSCLKOUT/128
3-0	UPPS	R/W	0h	Unit position event prescaler Reset type: SYSRSn 0h (R/W) = UPEVNT = QCLK/1 1h (R/W) = UPEVNT = QCLK/2 2h (R/W) = UPEVNT = QCLK/4 3h (R/W) = UPEVNT = QCLK/8 4h (R/W) = UPEVNT = QCLK/16 5h (R/W) = UPEVNT = QCLK/32 6h (R/W) = UPEVNT = QCLK/64 7h (R/W) = UPEVNT = QCLK/128 8h (R/W) = UPEVNT = QCLK/256 9h (R/W) = UPEVNT = QCLK/512 Ah (R/W) = UPEVNT = QCLK/1024 Bh (R/W) = UPEVNT = QCLK/2048 Ch (R/W) = Reserved Dh (R/W) = Reserved Eh (R/W) = Reserved Fh (R/W) = Reserved

### 21.11.2.15 QPOSCTL Register (Offset = 17h) [reset = 0h]

QPOSCTL is shown in [Figure 21-39](#) and described in [Table 21-20](#).

Return to the [Summary Table](#).

Position Compare Control

**Figure 21-39. QPOSCTL Register**

15		14		13		12		11		10		9		8	
PCSHDW		PCLOAD		PCPOL		PCE				PCSPW					
R/W-0h		R/W-0h		R/W-0h		R/W-0h				R/W-0h					
7		6		5		4		3		2		1		0	
										PCSPW					
										R/W-0h					

**Table 21-20. QPOSCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	PCSHDW	R/W	0h	Position compare of shadow enable Reset type: SYSRSn 0h (R/W) = Shadow disabled, load Immediate 1h (R/W) = Shadow enabled
14	PCLOAD	R/W	0h	Position compare of shadow load Reset type: SYSRSn 0h (R/W) = Load on QPOSCNT = 0 1h (R/W) = Load when QPOSCNT = QPOSCMP
13	PCPOL	R/W	0h	Polarity of sync output Reset type: SYSRSn 0h (R/W) = Active HIGH pulse output 1h (R/W) = Active LOW pulse output
12	PCE	R/W	0h	Position compare enable/disable Reset type: SYSRSn 0h (R/W) = Disable position compare unit 1h (R/W) = Enable position compare unit
11-0	PCSPW	R/W	0h	Select-position-compare sync output pulse width Reset type: SYSRSn 0h (R/W) = 1 * 4 * SYSCLKOUT cycles 1h (R/W) = 2 * 4 * SYSCLKOUT cycles FFFh (R/W) = 4096 * 4 * SYSCLKOUT cycles

**21.11.2.16 QEINT Register (Offset = 18h) [reset = 0h]**

QEINT is shown in [Figure 21-40](#) and described in [Table 21-21](#).

Return to the [Summary Table](#).

QEP Interrupt Control

**Figure 21-40. QEINT Register**

15	14	13	12	11	10	9	8
RESERVED			QMAE	UTO	IEL	SEL	PCM
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	QPE	PCE	RESERVED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

**Table 21-21. QEINT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	QMAE	R/W	0h	QMA Error Interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
11	UTO	R/W	0h	Unit time out interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
10	IEL	R/W	0h	Index event latch interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
9	SEL	R/W	0h	Strobe event latch interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
8	PCM	R/W	0h	Position-compare match interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
7	PCR	R/W	0h	Position-compare ready interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
6	PCO	R/W	0h	Position counter overflow interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
5	PCU	R/W	0h	Position counter underflow interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled

**Table 21-21. QEINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	WTO	R/W	0h	Watchdog time out interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
3	QDC	R/W	0h	Quadrature direction change interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
2	QPE	R/W	0h	Quadrature phase error interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
1	PCE	R/W	0h	Position counter error interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
0	RESERVED	R	0h	Reserved

**21.11.2.17 QFLG Register (Offset = 19h) [reset = 0h]**

QFLG is shown in [Figure 21-41](#) and described in [Table 21-22](#).

Return to the [Summary Table](#).

QEP Interrupt Flag

**Figure 21-41. QFLG Register**

15	14	13	12	11	10	9	8
RESERVED			QMAE	UTO	IEL	SEL	PCM
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 21-22. QFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	QMAE	R	0h	QMA Error interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
11	UTO	R	0h	Unit time out interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Set by eQEP unit timer period match
10	IEL	R	0h	Index event latch interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set after latching the QPOSCNT to QPOSSILAT
9	SEL	R	0h	Strobe event latch interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set after latching the QPOSCNT to QPOSSLAT
8	PCM	R	0h	eQEP compare match event interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set on position-compare match
7	PCR	R	0h	Position-compare ready interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set after transferring the shadow register value to the active position compare register
6	PCO	R	0h	Position counter overflow interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set on position counter overflow.
5	PCU	R	0h	Position counter underflow interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set on position counter underflow.

**Table 21-22. QFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	WTO	R	0h	Watchdog timeout interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Set by watchdog timeout
3	QDC	R	0h	Quadrature direction change interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
2	PHE	R	0h	Quadrature phase error interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Set on simultaneous transition of QEPA and QEPB
1	PCE	R	0h	Position counter error interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Position counter error
0	INT	R	0h	Global interrupt status flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated

**21.11.2.18 QCLR Register (Offset = 1Ah) [reset = 0h]**

QCLR is shown in [Figure 21-42](#) and described in [Table 21-23](#).

Return to the [Summary Table](#).

QEP Interrupt Clear

**Figure 21-42. QCLR Register**

15	14	13	12	11	10	9	8
RESERVED			QMAE	UTO	IEL	SEL	PCM
R-0h			R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 21-23. QCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	QMAE	R-0/W1S	0h	Clear QMA Error interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
11	UTO	R-0/W1S	0h	Clear unit time out interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
10	IEL	R-0/W1S	0h	Clear index event latch interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
9	SEL	R-0/W1S	0h	Clear strobe event latch interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
8	PCM	R-0/W1S	0h	Clear eQEP compare match event interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
7	PCR	R-0/W1S	0h	Clear position-compare ready interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
6	PCO	R-0/W1S	0h	Clear position counter overflow interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
5	PCU	R-0/W1S	0h	Clear position counter underflow interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag



**Table 21-23. QCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	WTO	R-0/W1S	0h	Clear watchdog timeout interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
3	QDC	R-0/W1S	0h	Clear quadrature direction change interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
2	PHE	R-0/W1S	0h	Clear quadrature phase error interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
1	PCE	R-0/W1S	0h	Clear position counter error interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
0	INT	R-0/W1S	0h	Global interrupt clear flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag

**21.11.2.19 QFRC Register (Offset = 1Bh) [reset = 0h]**

QFRC is shown in [Figure 21-43](#) and described in [Table 21-24](#).

Return to the [Summary Table](#).

QEP Interrupt Force

**Figure 21-43. QFRC Register**

15	14	13	12	11	10	9	8
RESERVED			QMAE	UTO	IEL	SEL	PCM
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	RESERVED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

**Table 21-24. QFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	QMAE	R/W	0h	Force QMA error interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
11	UTO	R/W	0h	Force unit time out interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
10	IEL	R/W	0h	Force index event latch interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
9	SEL	R/W	0h	Force strobe event latch interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
8	PCM	R/W	0h	Force position-compare match interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
7	PCR	R/W	0h	Force position-compare ready interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
6	PCO	R/W	0h	Force position counter overflow interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
5	PCU	R/W	0h	Force position counter underflow interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt

**Table 21-24. QFRC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	WTO	R/W	0h	Force watchdog time out interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
3	QDC	R/W	0h	Force quadrature direction change interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
2	PHE	R/W	0h	Force quadrature phase error interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
1	PCE	R/W	0h	Force position counter error interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
0	RESERVED	R	0h	Reserved

**21.11.2.20 QEPSTS Register (Offset = 1Ch) [reset = 80h]**

 QEPSTS is shown in [Figure 21-44](#) and described in [Table 21-25](#).

 Return to the [Summary Table](#).

QEP Status

**Figure 21-44. QEPSTS Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
UPEVNT	FIDF	QDF	QDLF	COEF	CDEF	FIMF	PCEF
R/W1S-1h	R-0h	R-0h	R-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R-0h

**Table 21-25. QEPSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	UPEVNT	R/W1S	1h	Unit position event flag Reset type: SYSRSn 0h (R/W) = No unit position event detected 1h (R/W) = Unit position event detected. Write 1 to clear
6	FIDF	R	0h	Direction on the first index marker Status of the direction is latched on the first index event marker. Reset type: SYSRSn 0h (R/W) = Counter-clockwise rotation (or reverse movement) on the first index event 1h (R/W) = Clockwise rotation (or forward movement) on the first index event
5	QDF	R	0h	Quadrature direction flag Reset type: SYSRSn 0h (R/W) = Counter-clockwise rotation (or reverse movement) 1h (R/W) = Clockwise rotation (or forward movement)
4	QDLF	R	0h	eQEP direction latch flag Reset type: SYSRSn 0h (R/W) = Counter-clockwise rotation (or reverse movement) on index event marker 1h (R/W) = Clockwise rotation (or forward movement) on index event marker
3	COEF	R/W1S	0h	Capture overflow error flag Reset type: SYSRSn 0h (R/W) = Overflow has not occurred. 1h (R/W) = Overflow occurred in eQEP Capture timer (QEPCTMR). This bit is cleared by writing a '1'.
2	CDEF	R/W1S	0h	Capture direction error flag Reset type: SYSRSn 0h (R/W) = Capture direction error has not occurred. 1h (R/W) = Direction change occurred between the capture position event. This bit is cleared by writing a '1'.
1	FIMF	R/W1S	0h	First index marker flag Reset type: SYSRSn 0h (R/W) = First index pulse has not occurred. 1h (R/W) = Set by first occurrence of index pulse. This bit is cleared by writing a '1'.

**Table 21-25. QEPSTS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	PCEF	R	0h	Position counter error flag. This bit is not sticky and it is updated for every index event. Reset type: SYSRSn 0h (R/W) = No error occurred during the last index transition 1h (R/W) = Position counter error

**21.11.2.21 QCTMR Register (Offset = 1Dh) [reset = 0h]**

QCTMR is shown in [Figure 21-45](#) and described in [Table 21-26](#).

Return to the [Summary Table](#).

QEP Capture Timer

**Figure 21-45. QCTMR Register**

15	14	13	12	11	10	9	8
QCTMR							
R/W-0h							
7	6	5	4	3	2	1	0
QCTMR							
R/W-0h							

**Table 21-26. QCTMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QCTMR	R/W	0h	This register provides time base for edge capture unit. Reset type: SYSRSn

**21.11.2.22 QCPRD Register (Offset = 1Eh) [reset = 0h]**

QCPRD is shown in [Figure 21-46](#) and described in [Table 21-27](#).

Return to the [Summary Table](#).

QEP Capture Period

**Figure 21-46. QCPRD Register**

15	14	13	12	11	10	9	8
QCPRD							
R/W-0h							
7	6	5	4	3	2	1	0
QCPRD							
R/W-0h							

**Table 21-27. QCPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QCPRD	R/W	0h	This register holds the period count value between the last successive eQEP position events Reset type: SYSRSn

**21.11.2.23 QCTMRLAT Register (Offset = 1Fh) [reset = 0h]**

QCTMRLAT is shown in [Figure 21-47](#) and described in [Table 21-28](#).

Return to the [Summary Table](#).

QEP Capture Latch

**Figure 21-47. QCTMRLAT Register**

15	14	13	12	11	10	9	8
QCTMRLAT							
R-0h							
7	6	5	4	3	2	1	0
QCTMRLAT							
R-0h							

**Table 21-28. QCTMRLAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QCTMRLAT	R	0h	The eQEP capture timer value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter. Reset type: SYSRSn



**21.11.2.24 QCPRDLAT Register (Offset = 20h) [reset = 0h]**

QCPRDLAT is shown in [Figure 21-48](#) and described in [Table 21-29](#).

Return to the [Summary Table](#).

QEP Capture Period Latch

**Figure 21-48. QCPRDLAT Register**

15	14	13	12	11	10	9	8
QCPRDLAT							
R-0h							
7	6	5	4	3	2	1	0
QCPRDLAT							
R-0h							

**Table 21-29. QCPRDLAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QCPRDLAT	R	0h	eQEP capture period value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter. Reset type: SYSRSn

**21.11.2.25 REV Register (Offset = 30h) [reset = 1h]**

REV is shown in [Figure 21-49](#) and described in [Table 21-30](#).

Return to the [Summary Table](#).

QEP Revision Number

**Figure 21-49. REV Register**

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

**Table 21-30. REV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-3	MINOR	R	0h	This field specifies the Minor Revision number for the eQEP IP. Reset type: N/A
2-0	MAJOR	R	1h	This field specifies the Major Revision number for the eQEP IP. Reset type: N/A

**21.11.2.26 QEPSTROBESEL Register (Offset = 32h) [reset = 0h]**

QEPSTROBESEL is shown in [Figure 21-50](#) and described in [Table 21-31](#).

Return to the [Summary Table](#).

QEP Strobe select register

**Figure 21-50. QEPSTROBESEL Register**

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

**Table 21-31. QEPSTROBESEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1-0	STROBESEL	R/W	0h	Strobe source select: Reset type: SYSRSn 0h (R/W) = QEP Strobe after polarity mux 1h (R/W) = QEP Strobe after polarity mux 2h (R/W) = QEP Strobe after polarity mux ORed with ADCSOCA 3h (R/W) = QEP Strobe after polarity mux ORed with ADCSOCB

**21.11.2.27 QMACTRL Register (Offset = 34h) [reset = 0h]**

QMACTRL is shown in [Figure 21-51](#) and described in [Table 21-32](#).

Return to the [Summary Table](#).

QMA Control register

**Figure 21-51. QMACTRL Register**

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

**Table 21-32. QMACTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2-0	MODE	R/W	0h	Select Mode for QMA mode: 000 : QMA Module is bypassed. 001 : QMA Mode-1 operation selected 010 : QMA Mode-2 operation selected 011 : QMA Module is bypassed (reserved) 1xx : QMA Module is bypassed (reserved) Reset type: SYSRSn

### 21.11.3 Register to Driverlib Function Mapping

**Table 21-33. EQEP Registers to Driverlib Functions**

File	Driverlib Function
<b>QPOSCNT</b>	
eqep.h	EQEP_getPosition
eqep.h	EQEP_setPosition
<b>QPOSINIT</b>	
eqep.h	EQEP_setInitialPosition
<b>QPOSMAX</b>	
eqep.h	EQEP_setPositionCounterConfig
<b>QPOSCMP</b>	
eqep.c	EQEP_setCompareConfig
<b>QPOSILAT</b>	
eqep.h	EQEP_getIndexPositionLatch
<b>QPOSSLAT</b>	
eqep.h	EQEP_getStrobePositionLatch
<b>QPOSLAT</b>	
eqep.h	EQEP_getPositionLatch
<b>QUPRD</b>	
eqep.h	EQEP_loadUnitTimer
eqep.h	EQEP_enableUnitTimer
<b>QWDTMR</b>	
eqep.h	EQEP_setWatchdogTimerValue
eqep.h	EQEP_getWatchdogTimerValue
<b>QWDPRD</b>	
eqep.h	EQEP_enableWatchdog
<b>QDECCTL</b>	
eqep.c	EQEP_setCompareConfig
eqep.c	EQEP_setInputPolarity
eqep.h	EQEP_setDecoderConfig
<b>QEPCTL</b>	
eqep.h	EQEP_enableModule
eqep.h	EQEP_disableModule
eqep.h	EQEP_setPositionCounterConfig
eqep.h	EQEP_enableUnitTimer
eqep.h	EQEP_disableUnitTimer
eqep.h	EQEP_enableWatchdog
eqep.h	EQEP_disableWatchdog
eqep.h	EQEP_setPositionInitMode
eqep.h	EQEP_setSWPositionInit
eqep.h	EQEP_setLatchMode
eqep.h	EQEP_setEmulationMode
<b>QCAPCTL</b>	
eqep.h	EQEP_setCaptureConfig
eqep.h	EQEP_enableCapture
eqep.h	EQEP_disableCapture
<b>QPOSCTL</b>	
eqep.c	EQEP_setCompareConfig
eqep.h	EQEP_enableCompare

**Table 21-33. EQEP Registers to Driverlib Functions (continued)**

File	Driverlib Function
eqep.h	EQEP_disableCompare
eqep.h	EQEP_setComparePulseWidth
<b>QEINT</b>	
eqep.h	EQEP_enableInterrupt
eqep.h	EQEP_disableInterrupt
<b>QFLG</b>	
eqep.h	EQEP_getInterruptStatus
eqep.h	EQEP_getError
<b>QCLR</b>	
eqep.h	EQEP_clearInterruptStatus
<b>QFRC</b>	
eqep.h	EQEP_forceInterrupt
<b>QEPSTS</b>	
eqep.h	EQEP_getDirection
eqep.h	EQEP_getStatus
eqep.h	EQEP_clearStatus
<b>QCTMR</b>	
eqep.h	EQEP_getCaptureTimer
eqep.h	EQEP_getCaptureTimerLatch
<b>QCPRD</b>	
eqep.h	EQEP_getCapturePeriod
eqep.h	EQEP_getCapturePeriodLatch
<b>QCTMRLAT</b>	
eqep.h	EQEP_getCaptureTimerLatch
<b>QCPRDLAT</b>	
eqep.h	EQEP_getCapturePeriodLatch
<b>QEPSTROBESEL</b>	
eqep.h	EQEP_setStrobeSource
<b>QMACTRL</b>	
eqep.h	EQEP_setQMAModuleMode

## Serial Peripheral Interface (SPI)

---

---

This chapter describes the serial peripheral interface (SPI) which is a high-speed synchronous serial input and output (I/O) port that allows a serial bit stream of programmed length (one to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communications between the MCU controller and external peripherals or another controller. Typical applications include external I/O or peripheral expansion via devices such as shift registers, display drivers, and analog-to-digital converters (ADCs). Multi-device communications are supported by the master or slave operation of the SPI. The port supports a 16-level, receive and transmit FIFO for reducing CPU servicing overhead.

Topic	Page
<b>22.1 Introduction</b> .....	<b>2208</b>
<b>22.2 System-Level Integration</b> .....	<b>2209</b>
<b>22.3 SPI Operation</b> .....	<b>2212</b>
<b>22.4 Programming Procedure</b> .....	<b>2222</b>
<b>22.5 SPI Registers</b> .....	<b>2227</b>

## 22.1 Introduction

### 22.1.1 Features

The SPI module features include:

- SPISOMI: SPI slave-output/master-input pin
- SPISIMO: SPI slave-input/master-output pin
- $\overline{\text{SPISTE}}$ : SPI slave transmit-enable pin
- SPICLK: SPI serial-clock pin

---

**NOTE:** All four pins can be used as GPIO if the SPI module is not used.

---

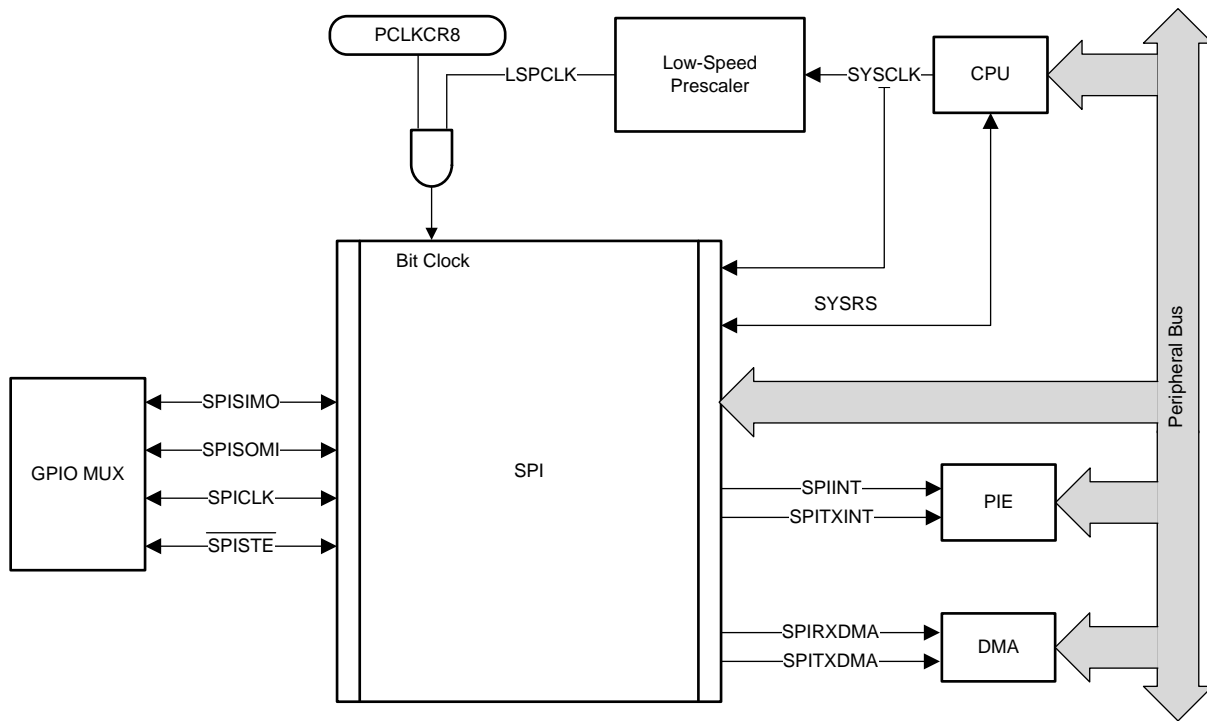
- Two operational modes: Master and Slave
- Baud rate: 125 different programmable rates. The maximum baud rate that can be employed is limited by the maximum speed of the I/O buffers used on the SPI pins. See the device-specific data manual for more details.
- Data word length: one to sixteen data bits
- Four clocking schemes (controlled by clock polarity and clock phase bits) include:
  - Falling edge without phase delay: SPICLK active-high. SPI transmits data on the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
  - Falling edge with phase delay: SPICLK active-high. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge without phase delay: SPICLK inactive-low. SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge with phase delay: SPICLK inactive-low. SPI transmits data one half-cycle ahead of the rising edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
- Simultaneous receive and transmit operation (transmit function can be disabled in software)
- Transmitter and receiver operations are accomplished through either interrupt- driven or polled algorithm
- 16-level transmit/receive FIFO
- DMA support
- High-speed mode
- Delayed transmit control
- 3-wire SPI mode
- $\overline{\text{SPISTE}}$  inversion for digital audio interface receive mode on devices with two SPI modules

### 22.1.2 Block Diagram

Figure 22-1 shows the SPI CPU interfaces.



Figure 22-1. SPI CPU Interface



## 22.2 System-Level Integration

This section describes the various functionality that is applicable to the device integration. These features require configuration of other modules in the device that are not within the scope of this chapter.

### 22.2.1 SPI Module Signals

Table 22-1 classifies and provides a summary of the SPI module signals.

Table 22-1. SPI Module Signal Summary

Signal Name	Description
<b>External Signals</b>	
SPICLK	SPI clock
SPISIMO	SPI slave in, master out
SPISOMI	SPI slave out, master in
$\overline{\text{SPISTE}}$	SPI slave transmit enable
<b>Control</b>	
SPI Clock Rate	LSPCLK
<b>Interrupt Signals</b>	
SPIINT/SPIRXINT	Transmit interrupt/ Receive Interrupt in non FIFO mode (referred to as SPIINT) Receive interrupt in FIFO mode
SPITXINT	Transmit interrupt in FIFO mode
<b>DMA Triggers</b>	
SPITXDMA	Transmit request to DMA
SPIRXDMA	Receive request to DMA

## Special Considerations

The  $\overline{\text{SPISTE}}$  signal provides the ability to gate any spurious clock and data pulses when the SPI is in slave mode. An active  $\overline{\text{SPISTE}}$  will not allow the slave to receive data. This prevents the SPI slave from losing synchronization with the master. It is this reason that TI does not recommend that the  $\overline{\text{SPISTE}}$  always be tied to the active state.

If the SPI slave does ever lose synchronization with the master, toggling  $\text{SPISWRESET}$  will reset internal bit counter as well as the various status flags in the module. By resetting the bit counter, the SPI will interpret the next clock transition as the first bit of a new transmission. The register bit fields which are reset by  $\text{SPISWRESET}$  can be found in [Section 22.5](#)

## Configuring a GPIO to emulate $\overline{\text{SPISTE}}$

In many systems, a SPI master may be connected to multiple SPI slaves using multiple instances of  $\overline{\text{SPISTE}}$ . Though this SPI module does not natively support multiple  $\overline{\text{SPISTE}}$  signals, it is possible to emulate this behavior in software using GPIOs. In this configuration, the SPI must be configured as the master. Rather than using the GPIO Mux to select  $\overline{\text{SPISTE}}$ , the application would configure pins to be GPIO outputs, one GPIO per SPI slave. Before transmitting any data, the application would drive the desired GPIO to the active state. Immediately after the transmission has been completed, the GPIO chip select would be driven to the inactive state. This process can be repeated for many slaves which share the  $\text{SPICLK}$ ,  $\text{SPISIMO}$ , and  $\text{SPISOMI}$  lines.

## 22.2.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the  $\text{GPyGMUX}$  bits must be configured first (while keeping the corresponding  $\text{GPyMUX}$  bits at the default of zero), followed by writing the  $\text{GPyMUX}$  register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate  $\text{GPxQSELn}$  register bits to 11b. The internal pullups can be configured in the  $\text{GPyPUD}$  register.

See the *GPIO* chapter for more details on GPIO mux and settings.

### 22.2.2.1 GPIOs Required for High-Speed Mode

The high-speed mode of the SPI is available on all GPIO mux options. To enable the high-speed enhancements, set  $\text{SPICCR.HS\_MODE}$  to 1. Ensure that the capacitive loading on the pin does not exceed the value stated in the device Data Manual.

When not operating in high-speed mode, or if the capacitive loading on the pins exceed the value stated in the device Data Manual,  $\text{SPICCR.HS\_MODE}$  should be set to 0.

## 22.2.3 SPI Interrupts

This section includes information on the available interrupts present in the SPI module.

The SPI module contains two interrupt lines:  $\text{SPIINT}/\text{SPIRXINT}$  and  $\text{SPITXINT}$ . When the SPI is operating in non-FIFO mode, all available interrupts are routed together to generate the single  $\text{SPIINT}$  interrupt. When FIFO mode is used, both  $\text{SPIRXINT}$  and  $\text{SPITXINT}$  can be generated.

### **SPIINT/SPIRXINT**

When the SPI is operating in non-FIFO mode, the interrupt generated is called  $\text{SPIINT}$ . If FIFO enhancements are enabled, the interrupt is called  $\text{SPIRXINT}$ . These interrupts share the same interrupt vector in the Peripheral Interrupt Expansion (PIE) block.

In non-FIFO mode, two conditions can trigger an interrupt: a transmission is complete ( $\text{INT\_FLAG}$ ), or there is overrun in the receiver ( $\text{OVERRUN\_FLAG}$ ). Both of these conditions share the same interrupt vector:  $\text{SPIINT}$ .

The transmission complete flag ( $\text{INT\_FLAG}$ ) indicates that the SPI has completed sending or receiving the last bit and is ready to be serviced. At the same time this bit is set, the received character is placed in the receiver buffer ( $\text{SPIRXBUF}$ ). The  $\text{INT\_FLAG}$  will generate an interrupt on the  $\text{SPIINT}$  vector if the  $\text{SPIINTENA}$  bit is set.

The receiver overrun flag (OVERRUN\_FLAG) indicates that a transmit or receive operation has completed before the previous character has been read from the buffer. The OVERRUN\_FLAG will generate an interrupt on the SPIINT vector if the OVERRUNINTENA bit is set and OVERRUN\_FLAG was previously cleared.

In FIFO mode, the SPI can interrupt the CPU upon a match condition between the current receive FIFO status (RXFFST) and the receive FIFO interrupt level (RXFFIL). If RXFFST is greater than or equal to RXFFIL, the receive FIFO interrupt flag (RXFFINT) will be set. SPIRXINT will be triggered in the PIE block if RXFFINT is set and the receive FIFO interrupt is enabled (RXFFIENA = 1).

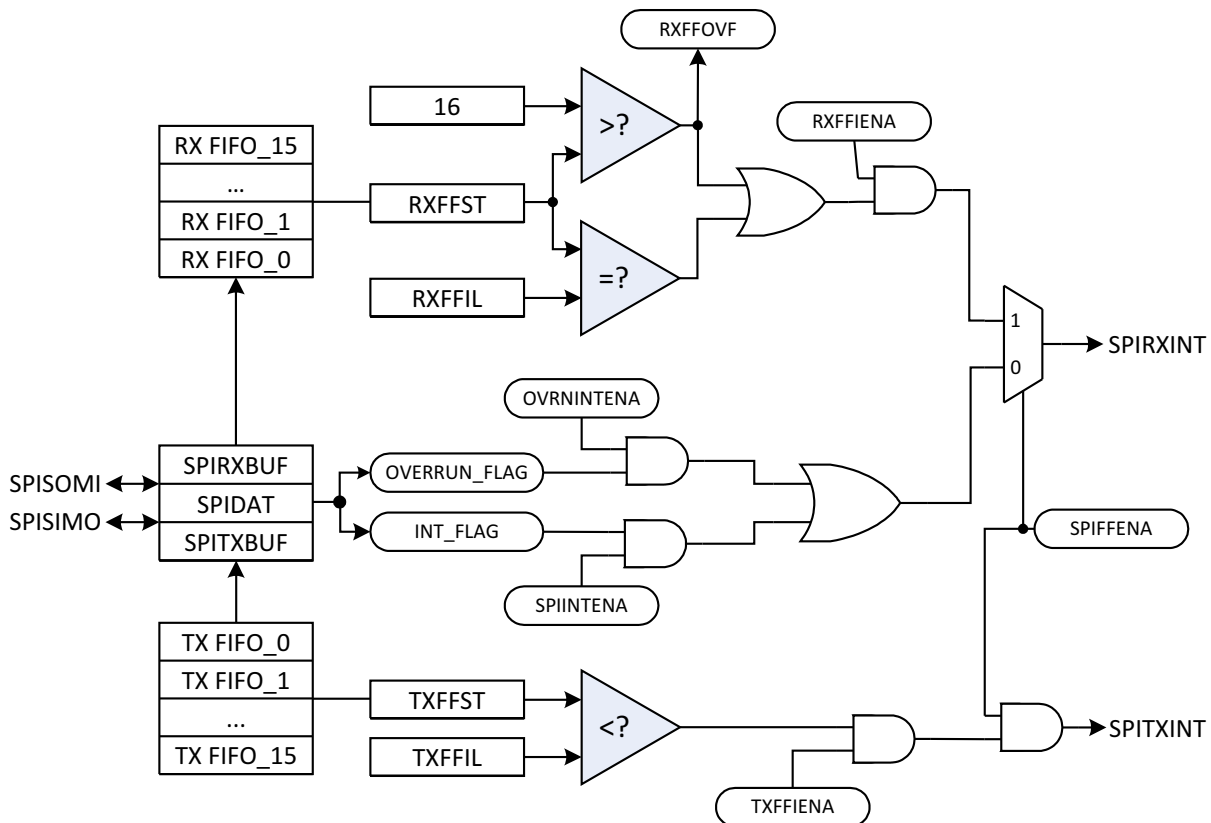
**SPITXINT**

The SPITXINT interrupt is not available when the SPI is operating in non-FIFO mode.

In FIFO mode, the SPITXINT behavior is similar to the SPIRXINT. SPITXINT is generated upon a match condition between the current transmit FIFO status (TXFFST) and the transmit FIFO interrupt level (TXFFIL). If TXFFST is less than or equal to TXFFIL, the transmit FIFO interrupt flag (TXFFINT) will be set. SPITXINT will be triggered in the PIE block if TXFFINT is set and the transmit FIFO interrupt is enabled in the SPI module (TXFFIENA = 1).

Figure 22-2 and Table 22-2 show how these control bits influence the SPI interrupt generation.

**Figure 22-2. SPI Interrupt Flags and Enable Logic Generation**



**Table 22-2. SPI Interrupt Flag Modes**

FIFO Options	SPI interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable (SPIFFENA)	Interrupt <sup>(1)</sup> Line
SPI without FIFO	Receive overrun	RXOVRN	OVRNINTENA	0	SPIRXINT
	Data receive	SPIINT	SPIINTENA	0	SPIRXINT
	Transmit empty	SPIINT	SPIINTENA	0	SPIRXINT
SPI FIFO mode	FIFO receive	RXFFIL	RXFFIENA	1	SPIRXINT
	Transmit empty	TXFFIL	TXFFIENA	1	SPITXINT

<sup>(1)</sup> In non-FIFO mode, SPIRXINT is the same name as the SPIINT interrupt in 28x devices.

### 22.2.4 DMA Support

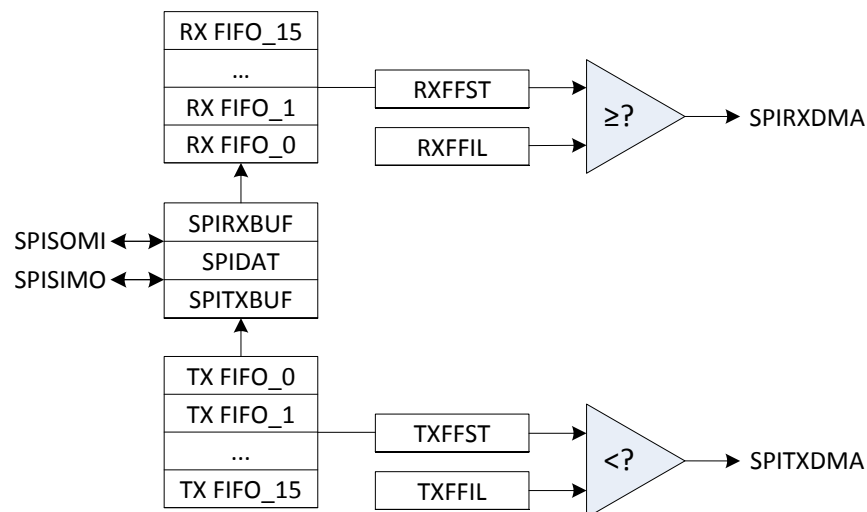
Both the CPU and DMA have access to the SPI data registers via the internal peripheral bus. This access is limited to 16-bit register read/writes. Each SPI module can generate two DMA events, SPITXDMA and SPIRXDMA. The DMA events are controlled by configuring the SPIFFTX.TXFFIL and SPIFFRX.RXFFIL appropriately. SPITXDMA activates when TXFFST is less than the interrupt level (TXFFIL). SPIRXDMA activates when RXFFST is greater than or equal to the interrupt level (RXFFIL).

The SPI must have FIFO enhancements enabled in order for the DMA triggers to be generated.

For more information on configuring the SPI for DMA transfers refer to [Section 22.3.8](#).

[Figure 22-3](#) is a block diagram showing the DMA trigger generation from the SPI module.

**Figure 22-3. SPI DMA Trigger Diagram**



## 22.3 SPI Operation

This section describes the various modes of operation of the SPI. Included are explanations of the operational modes, interrupts, data format, clock sources, and initialization. Typical timing diagrams for data transfers are given.

### 22.3.1 Introduction to Operation

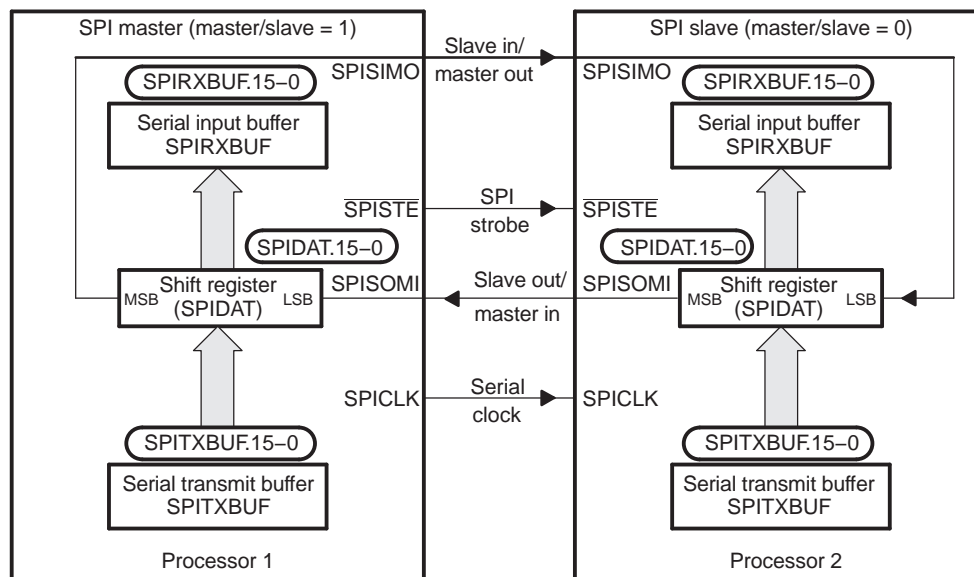
[Figure 22-4](#) shows typical connections of the SPI for communications between two controllers: a master and a slave.

The master transfers data by sending the SPICLK signal. For both the slave and the master, data is shifted out of the shift registers on one edge of the SPICLK and latched into the shift register on the opposite SPICLK clock edge. If the CLK\_PHASE bit is high, data is transmitted and received a half-cycle before the SPICLK transition. As a result, both controllers send and receive data simultaneously. The application software determines whether the data is meaningful or dummy data. There are three possible methods for data transmission:

- Master sends data; slave sends dummy data.
- Master sends data; slave sends data.
- Master sends dummy data; slave sends data.

The master can initiate data transfer at any time because it controls the SPICLK signal. The software, however, determines how the master detects when the slave is ready to broadcast data.

**Figure 22-4. SPI Master/Slave Connection**



The SPI can operate in master or slave mode. The MASTER\_SLAVE bit selects the operating mode and the source of the SPICLK signal.

### 22.3.2 Master Mode

In master mode (MASTER\_SLAVE = 1), the SPI provides the serial clock on the SPICLK pin for the entire serial communications network. Data is output on the SPISIMO pin and latched from the SPISOMI pin.

The SPIBRR register determines both the transmit and receive bit transfer rate for the network. SPIBRR can select 125 different data transfer rates.

Data written to SPIDAT or SPITXBUF initiates data transmission on the SPISIMO pin, MSB (most significant bit) first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB (least significant bit) of SPIDAT. When the selected number of bits has been transmitted, the received data is transferred to the SPIRXBUF (buffered receiver) for the CPU to read. Data is stored right-justified in SPIRXBUF.

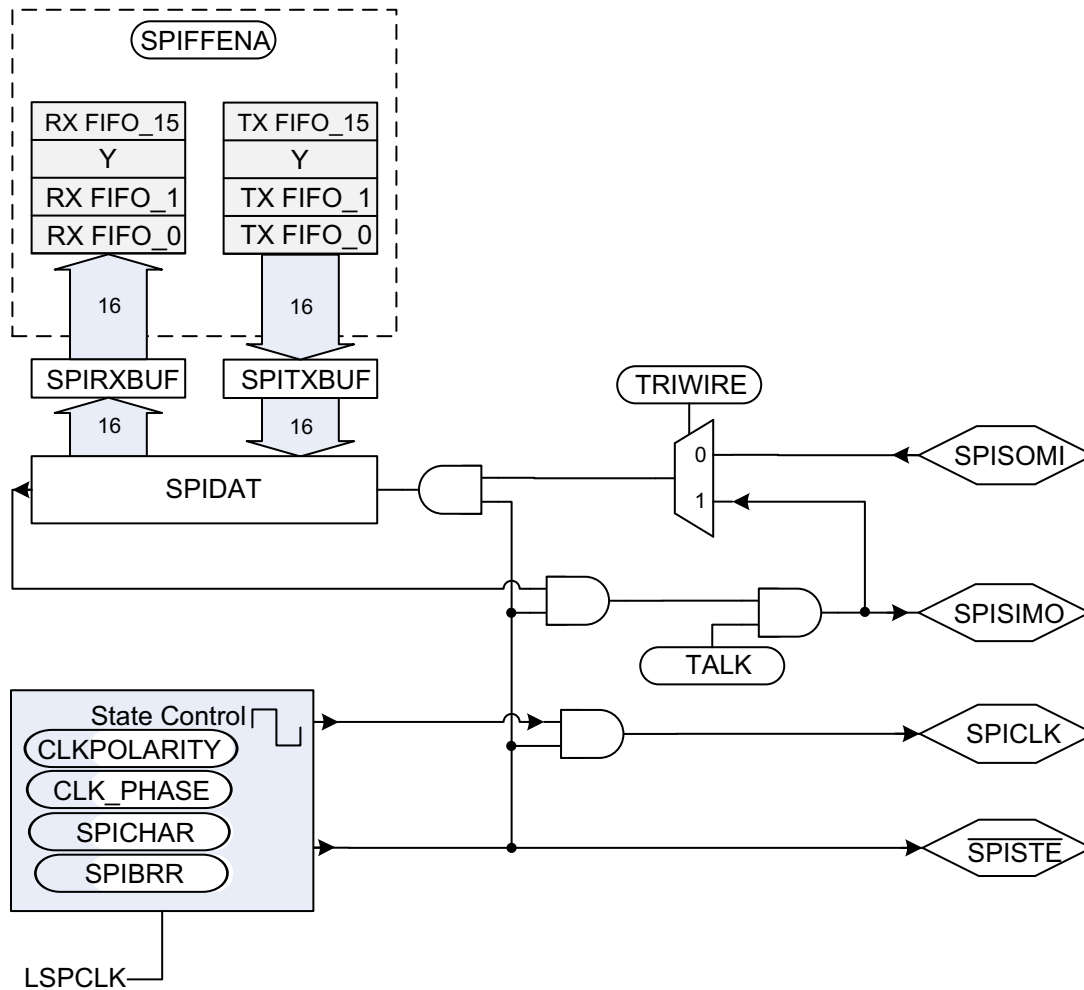
When the specified number of data bits has been shifted through SPIDAT, the following events occur:

- SPIDAT contents are transferred to SPIRXBUF.
- INT\_FLAG bit is set to 1.
- If there is valid data in the transmit buffer SPITXBUF, as indicated by the Transmit Buffer Full Flag (BUFFULL\_FLAG), this data is transferred to SPIDAT and is transmitted; otherwise, SPICLK stops after all bits have been shifted out of SPIDAT.
- If the SPIINTENA bit is set to 1, an interrupt is asserted.

In a typical application, the  $\overline{\text{SPISTE}}$  pin serves as a chip-enable pin for a slave SPI device. This pin is driven low by the master before transmitting data to the slave and is taken high after the transmission is complete.

Figure 22-5 is a block diagram of the SPI in master mode. It shows the basic control blocks available in SPI master mode.

**Figure 22-5. SPI Module Master Configuration**



### 22.3.3 Slave Mode

In slave mode ( $\text{MASTER\_SLAVE} = 0$ ), data shifts out on the SPISOMI pin and in on the SPISIMO pin. The SPICLK pin is used as the input for the serial shift clock, which is supplied from the external network master. The transfer rate is defined by this clock. The SPICLK input frequency should be no greater than the LSPCLK frequency divided by 4.

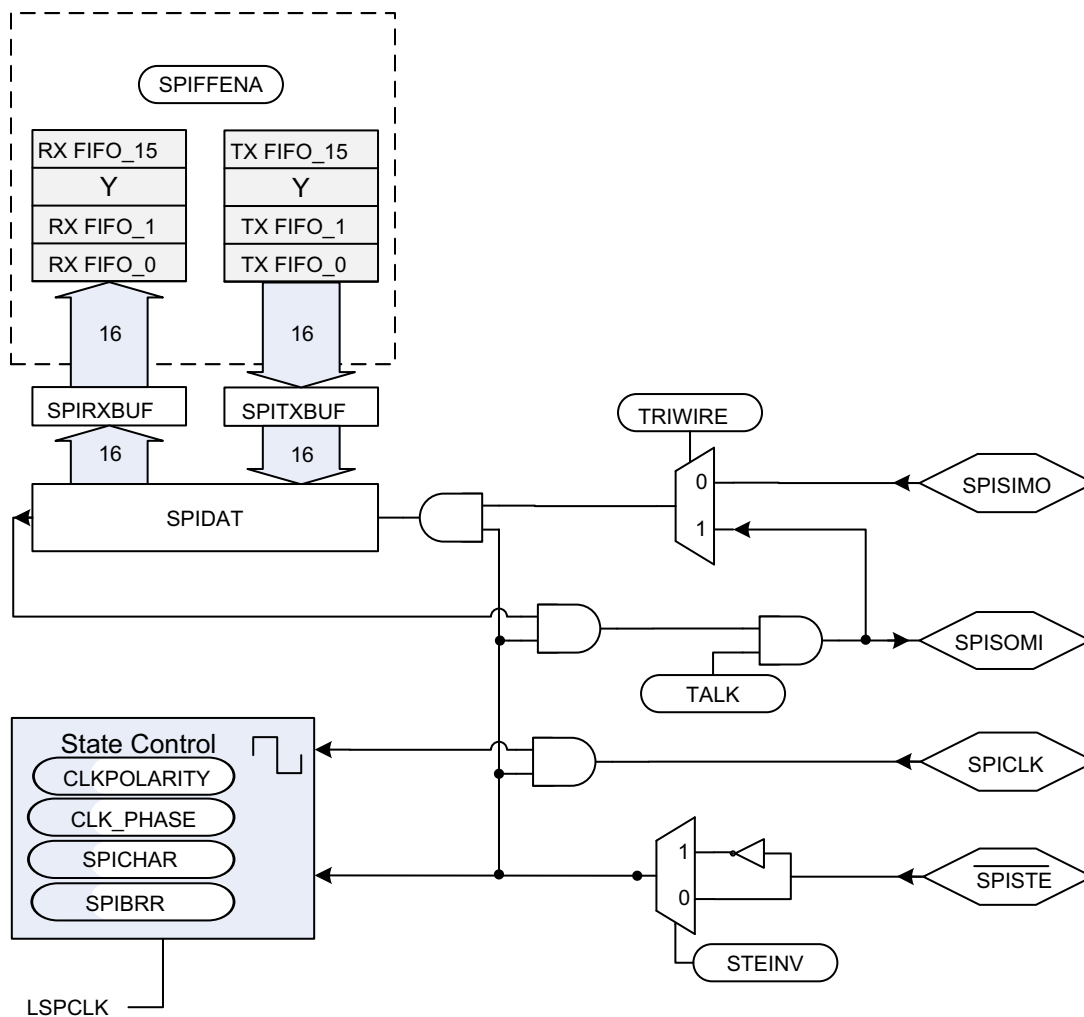
Data written to SPIDAT or SPITXBUF is transmitted to the network when appropriate edges of the SPICLK signal are received from the network master. Data written to the SPITXBUF register will be transferred to the SPIDAT register when all bits of the character to be transmitted have been shifted out of SPIDAT. If no character is currently being transmitted when SPITXBUF is written to, the data will be transferred immediately to SPIDAT. To receive data, the SPI waits for the network master to send the SPICLK signal and then shifts the data on the SPISIMO pin into SPIDAT. If data is to be transmitted by the slave simultaneously, and SPITXBUF has not been previously loaded, the data must be written to SPITXBUF or SPIDAT before the beginning of the SPICLK signal.

When the TALK bit is cleared, data transmission is disabled, and the output line (SPISOMI) is put into the high-impedance state. If this occurs while a transmission is active, the current character is completely transmitted even though SPISOMI is forced into the high-impedance state. This ensures that the SPI is still able to receive incoming data correctly. This TALK bit allows many slave devices to be tied together on the network, but only one slave at a time is allowed to drive the SPISOMI line.

The  $\overline{\text{SPISTE}}$  pin operates as the slave-select pin. An active-low signal on the  $\overline{\text{SPISTE}}$  pin allows the slave SPI to transfer data to the serial data line; an inactive-high signal causes the slave SPI serial shift register to stop and its serial output pin to be put into the high-impedance state. This allows many slave devices to be tied together on the network, although only one slave device is selected at a time.

Figure 22-6 is a block diagram of the SPI in slave mode. It shows the basic control blocks available in SPI slave mode.

Figure 22-6. SPI Module Slave Configuration



### 22.3.4 Data Format

The four-bit SPICHR register field specifies the number of bits in the data character (1 to 16). This information directs the state control logic to count the number of bits received or transmitted to determine when a complete character has been processed.

The following statements apply to characters with fewer than 16 bits:

- Data must be left-justified when written to SPIDAT and SPITXBUF.
- Data read back from SPIRXBUF is right-justified.
- SPIRXBUF contains the most recently received character, right-justified, plus any bits that remain from

previous transmission(s) that have been shifted to the left (shown in [Example 22-1](#)).

### Example 22-1. Transmission of Bit From SPIRXBUF

Conditions:

1. Transmission character length = 1 bit (specified in bits SPICHR)
2. The current value of SPIDAT = 737Bh

SPIDAT (before transmission)																	
	0	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	
SPIDAT (after transmission)																	
(TXed) 0 ←	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x <sup>(1)</sup>	← (RXed)
SPIRXBUF (after transmission)																	
	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x <sup>(1)</sup>	

<sup>(1)</sup> x = 1 if SPISOMI data is high; x = 0 if SPISOMI data is low; master mode is assumed.

### 22.3.5 Baud Rate Selection

The SPI module supports 125 different baud rates and four different clock schemes. Depending on whether the SPI clock is in slave or master mode, the SPICLK pin can receive an external SPI clock signal or provide the SPI clock signal, respectively.

- In the slave mode, the SPI clock is received on the SPICLK pin from the external source, and can be no greater than the LSPCLK frequency divided by 4.
- In the master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin, and can be no greater than the LSPCLK frequency divided by 4.

**NOTE:** The baud rate should be configured to not exceed the maximum rated GPIO toggle frequency. Refer to the device Data Manual for the maximum GPIO toggle frequency

[Example 22-2](#) shows how to determine the SPI baud rates.

### Example 22-2. Baud Rate Determination

For SPIBRR = 3 to 127:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)} \quad (8)$$

For SPIBRR = 0, 1, or 2:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4} \quad (9)$$

where:

LSPCLK = Low-speed peripheral clock frequency of the device

SPIBRR = Contents of the SPIBRR in the master SPI device

To determine what value to load into SPIBRR, you must know the device system clock (LSPCLK) frequency (which is device-specific) and the baud rate at which you will be operating.

The following example shows how to calculate the baud rate of the SPI module in standard SPI mode (HS\_MODE=0).



**Example 22-3. Baud Rate Calculation in Non-High Speed Mode (HS\_MODE=0)**

$$\begin{aligned}
 \text{SPI Baud Rate} &= \frac{\text{LSPCLK}}{\text{SPIBRR} + 1}, \quad \text{LSPCLK} = 50 \text{ MHz} \\
 &= \frac{50 \times 10^6}{3 + 1} \\
 &= 12.5 \text{ Mbps}
 \end{aligned}
 \tag{10}$$

**22.3.6 SPI Clocking Schemes**

The clock polarity select bit (CLKPOLARITY) and the clock phase select bit (CLK\_PHASE) control four different clocking schemes on the SPICLK pin. CLKPOLARITY selects the active edge, either rising or falling, of the clock. CLK\_PHASE selects a half-cycle delay of the clock. The four different clocking schemes are as follows:

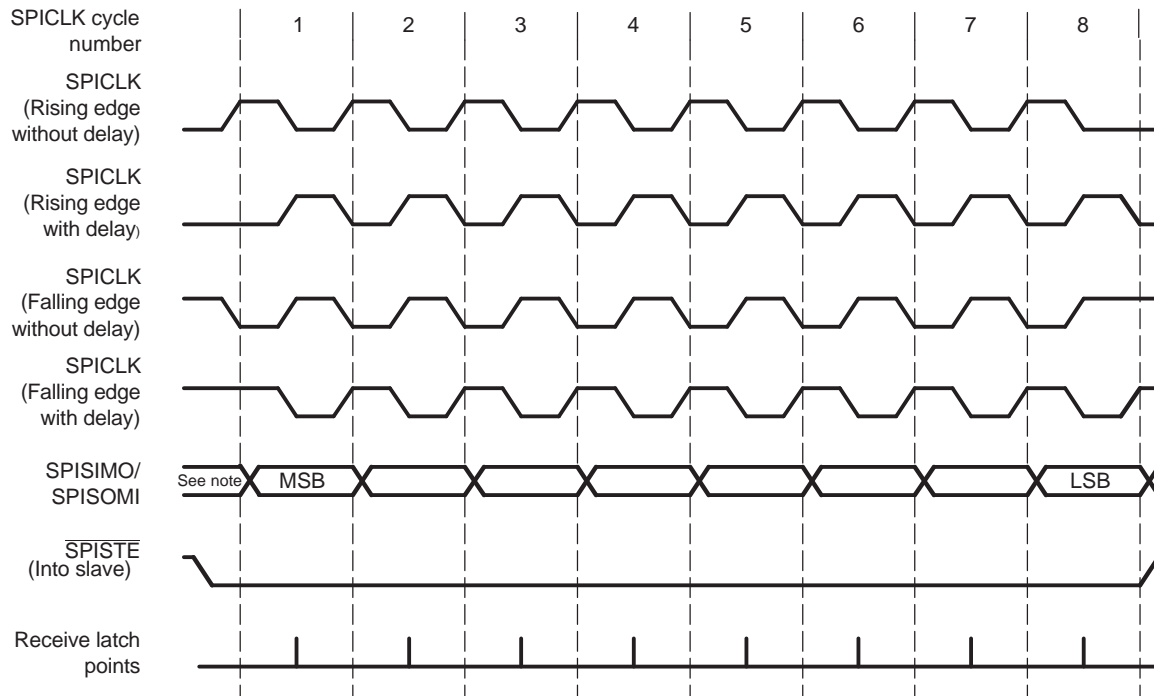
- **Falling Edge Without Delay.** The SPI transmits data on the falling edge of the SPICLK and receives data on the rising edge of the SPICLK.
- **Falling Edge With Delay.** The SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- **Rising Edge Without Delay.** The SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- **Rising Edge With Delay.** The SPI transmits data one half-cycle ahead of the rising edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.

The selection procedure for the SPI clocking scheme is shown in [Table 22-3](#). Examples of these four clocking schemes relative to transmitted and received data are shown in [Figure 22-7](#).

**Table 22-3. SPI Clocking Scheme Selection Guide**

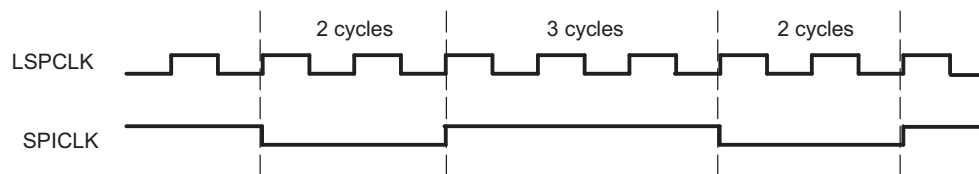
SPICLK Scheme	CLKPOLARITY	CLK_PHASE <sup>(1)</sup>
Rising edge without delay	0	0
Rising edge with delay	0	1
Falling edge without delay	1	0
Falling edge with delay	1	1

<sup>(1)</sup> The description of CLK\_PHASE and CLKPOLARITY differs between manufacturers. For proper operation, select the desired waveform to determine the clock phase and clock polarity settings.

**Figure 22-7. SPICLK Signal Options**


**Note:** Previous data bit

SPICLK symmetry is retained only when the result of  $(\text{SPIBRR}+1)$  is an even value. When  $(\text{SPIBRR} + 1)$  is an odd value and SPIBRR is greater than 3, SPICLK becomes asymmetrical. The low pulse of SPICLK is one LSPCLK cycle longer than the high pulse when CLKPOLARITY bit is clear (0). When CLKPOLARITY bit is set to 1, the high pulse of the SPICLK is one LSPCLK cycle longer than the low pulse, as shown in [Figure 22-8](#).

**Figure 22-8. SPI: SPICLK-LSPCLK Characteristic When  $(\text{BRR} + 1)$  is Odd,  $\text{BRR} > 3$ , and CLKPOLARITY = 1**


### 22.3.7 SPI FIFO Description

The following steps explain the FIFO features and help with programming the SPI FIFOs:

1. **Reset.** At reset the SPI powers up in standard SPI mode and the FIFO function is disabled. The FIFO registers SPIFFTX, SPIFFRX and SPIFFCT remain inactive.
2. **Standard SPI.** The standard 28x SPI mode will work with SPIINT/SPIRXINT as the interrupt source.
3. **Mode change.** FIFO mode is enabled by setting the SPIFFENA bit to 1 in the SPIFFTX register. SPIRST can reset the FIFO mode at any stage of its operation.
4. **Active registers.** All the SPI registers and SPI FIFO registers SPIFFTX, SPIFFRX, and SPIFFCT will be active.
5. **Interrupts.** FIFO mode has two interrupts one for the transmit FIFO, SPITXINT and one for the receive FIFO, SPIRXINT. SPIRXINT is the common interrupt for SPI FIFO receive, receive error and receive FIFO overflow conditions. The single SPIINT for both transmit and receive sections of the standard SPI will be disabled and this interrupt will service as SPI receive FIFO interrupt. For more information, refer to [Section 22.2.3](#)

6. **Buffers.** Transmit and receive buffers are each supplemented with a 16 word FIFO. The one-word transmit buffer (SPITXBUF) of the standard SPI functions as a transition buffer between the transmit FIFO and shift register. The one-word transmit buffer will be loaded from transmit FIFO only after the last bit of the shift register is shifted out.
7. **Delayed transfer.** The rate at which transmit words in the FIFO are transferred to transmit shift register is programmable. The SPIFFCT register bits (7–0) FFTXDLY7–FFTXDLY0 define the delay between the word transfer. The delay is defined in number SPI serial clock cycles. The 8-bit register could define a minimum delay of 0 SPICLK cycles and a maximum of 255 SPICLK cycles. With zero delay, the SPI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 255 clock delay, the SPI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 255 SPICLK cycles between each words. The programmable delay facilitates glueless interface to various slow SPI peripherals, such as EEPROMs, ADC, DAC, and so on.
8. **FIFO status bits.** Both transmit and receive FIFOs have status bits TXFFST or RXFFST that define the number of words available in the FIFOs at any time. The transmit FIFO reset bit (TXFIFO) and receive reset bit (RXFIFO) will reset the FIFO pointers to zero when these bits are set to 1. The FIFOs will resume operation from start once these bits are cleared to zero.
9. **Programmable interrupt levels.** Both transmit and receive FIFOs can generate CPU interrupts and DMA triggers. The transmit interrupt (SPITXINT) is generated whenever the transmit FIFO status bits (TXFFST) match (less than or equal to) the interrupt trigger level bits (TXFFIL). The receive interrupt (SPIRXINT) is generated whenever the receive FIFO status bits (RXFFST) match (greater than or equal to) the interrupt trigger level RXFFIL. This provides a programmable interrupt trigger for transmit and receive sections of the SPI. The default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO, respectively.

## 22.3.8 SPI DMA Transfers

### 22.3.8.1 Transmitting Data Using SPI with DMA

When using the DMA with the TX FIFO, the DMA Burst Size (DMA\_BURST\_SIZE) should be no greater than  $16 - \text{TXFFIL}$  in order to prevent the DMA from writing to an already full FIFO. This will lead to data loss and is not recommended.

For complete data transmission, please follow these steps:

1. Calculate the total number or words to be transmitted. [NUM\_WORDS]
2. Decide the transmit FIFO level. [TXFFIL]
3. Calculate the number of DMA transfers. [DMA\_TRANSFER\_SIZE]
4. Calculate the size of the DMA Burst. [DMA\_BURST\_SIZE]
5. Configure DMA using calculated values.
6. Configure SPI with FIFO using the calculated values.

To transfer 128 words to SPI using the DMA:

NUM\_WORDS: 128

TXFFIL: 8

DMA\_TRANSFER\_SIZE:  $(\text{NUM\_WORDS} / \text{TXFFIL}) - 1 = (128/8) - 1 = 15$  (16 transfers)

DMA\_BURST\_SIZE:  $(16 - \text{TXFFIL}) - 1 = (16 - 8) - 1 = 7$  (8 words per burst)

### 22.3.8.2 Receiving Data Using SPI with DMA

When using the DMA with the RX FIFO, the DMA Burst Size (BURST\_SIZE) should be no greater than RXFFIL in order to prevent the DMA from reading from an empty FIFO. To ensure that the DMA correctly receives all data from the RX FIFO, the DMA Burst Size should equal RXFFIL and also be an integer divisor of the total number of SPI transmissions.

For complete data reception, please follow these steps:

1. Calculate the number of words to be received. [NUM\_WORDS]
2. Calculate the necessary FIFO level [RXFFIL]

3. Calculate the total number of DMA transfers. [DMA\_TRANSFER\_SIZE]
4. Calculate the size of the DMA Burst. [DMA\_BURST\_SIZE]
5. Configure DMA using the calculated values.
6. Configure SPI with FIFO using the calculated values.

To receive 200 words from SPI using the DMA:

NUM\_WORDS = 200

RXFFIL: 4

DMA\_TRANSFER\_SIZE: (NUM\_WORDS / RXFFIL) – 1 = (200/4) – 1 = 49 (50 transfers)

DMA\_BURST\_SIZE = RXFFIL-1 = 3 (4 words per burst)

### 22.3.9 SPI High-Speed Mode

The SPI module is capable of reaching full-duplex communication speeds up to LSPCLK/4 (where LSPCLK equals SYSCLK). For the maximum rated speed, refer the device Data Manual.

In order to achieve the maximum full-duplex speeds, the following restrictions are placed on the design:

- Single master to single slave configuration is supported.
- Loading on the pins must not exceed the value stated in the device Data Manual.

When configuring the GPIOs to support High-Speed mode, refer to [Section 22.2.2.1](#) for more information.

### 22.3.10 SPI 3-Wire Mode Description

SPI 3-wire mode allows for SPI communication over three pins instead of the normal four pins.

In master mode, if the TRIWIRE bit is set, enabling 3-wire SPI mode, SPISIMOX becomes the bi-directional SPIMOMIx (SPI master out, master in) pin, and SPISOMIx is no longer used by the SPI. In slave mode, if the TRIWIRE bit is set, SPISOMIx becomes the bi-directional SPISISOx (SPI slave in, slave out) pin, and SPISIMOX is no longer used by the SPI.

[Table 22-4](#) indicates the pin function differences between 3-wire and 4-wire SPI mode for a master and slave SPI.

**Table 22-4. 4-wire vs. 3-wire SPI Pin Functions**

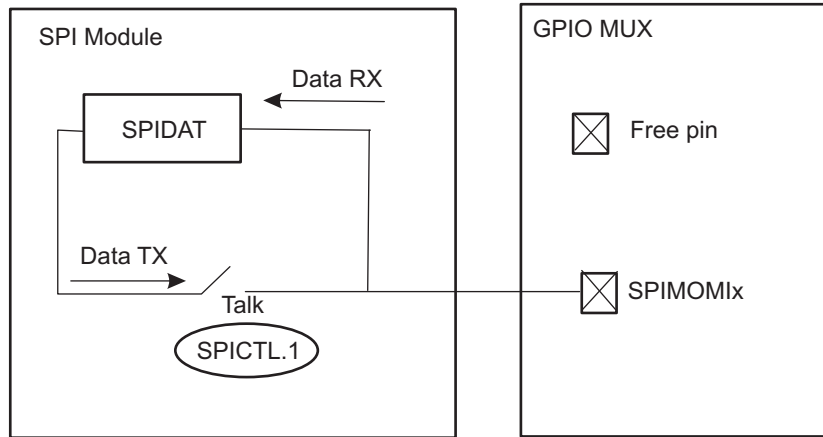
4-wire SPI	3-wire SPI (Master)	3-wire SPI(Slave)
SPICLKx	SPICLKx	SPICLKx
SPISTEx	SPISTEx	SPISTEx
SPISIMOX	SPIMOMIx	Free
SPISOMIx	Free	SPISISOx

Because in 3-wire mode, the receive and transmit paths within the SPI are connected, any data transmitted by the SPI module is also received by itself. The application software must take care to perform a dummy read to clear the SPI data register of the additional received data.

The TALK bit plays an important role in 3-wire SPI mode. The bit must be set to transmit data and cleared prior to reading data. In master mode, in order to initiate a read, the application software must write dummy data to the SPI data register (SPIDAT or SPIRXBUF) while the TALK bit is cleared (no data is transmitted out the SPIMOMI pin) before reading from the data register.

[Figure 22-9](#) and [Figure 22-10](#) illustrate 3-wire master and slave mode.

**Figure 22-9. SPI 3-wire Master Mode**



**Figure 22-10. SPI 3-wire Slave Mode**

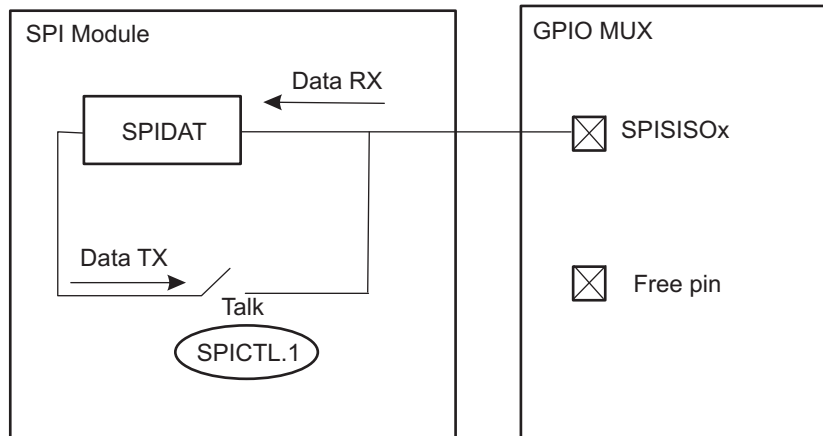


Table 22-5 indicates how data is received or transmitted in the various SPI modes while the TALK bit is set or cleared.

**Table 22-5. 3-Wire SPI Pin Configuration**

Pin Mode	SPIPRI[TRIWIRE]	SPICTL[TALK]	SPISIMO	SPISOMI
<b>Master Mode</b>				
4-wire	0	X	TX	RX
3-pin mode	1	0	RX	Disconnect from SPI
		1	TX/RX	
<b>Slave Mode</b>				
4-wire	0	X	RX	TX
3-pin mode	1	0	Disconnect from SPI	RX
		1		TX/RX

## 22.4 Programming Procedure

This section describes the procedure for configuring the SPI for the various modes of operation.

### 22.4.1 Initialization Upon Reset

A system reset forces the SPI peripheral into the following default configuration:

- Unit is configured as a slave module (MASTER\_SLAVE = 0)
- Transmit capability is disabled (TALK = 0)
- Data is latched at the input on the falling edge of the SPICLK signal
- Character length is assumed to be one bit
- SPI interrupts are disabled
- Data in SPIDAT is reset to 0000h

### 22.4.2 Configuring the SPI

This section describes the procedure in which to configure the SPI module for operation. To prevent unwanted and unforeseen events from occurring during or as a result of initialization changes, clear the SPISWRESET bit before making initialization changes, and then set this bit after initialization is complete. While the SPI is held in reset (SPISWRESET = 0), configuration may be changed in any order. The following list shows the SPI configuration procedure in a logical order. However, the SPI registers can be written with single 16-bit writes, so the order is not required with the exception of SPISWRESET.

To change the SPI configuration:

Step 1. Clear the SPI Software Reset bit (SPISWRESET) to 0 to force the SPI to the reset state.

Step 2. Configure the SPI as desired:

- Select either master or slave mode (MASTER\_SLAVE).
- Choose SPICLK polarity and phase (CLKPOLARITY and CLK\_PHASE).
- Set the desired baud rate (SPIBRR).
- Enable high speed mode if desired (HS\_MODE).
- Set the SPI character length (SPICHR).
- Clear the SPI Flags (OVERRUN\_FLAG, INT\_FLAG).
- Enable  $\overline{\text{SPISTE}}$  inversion (STEINV) if needed.
- Enable 3-wire mode (TRIWIRE) if needed.
- If using FIFO enhancements:
  - Enable the FIFO enhancements (SPIFFENA).
  - Clear the FIFO Flags (TXFFINTCLR, RXFFOVFCLR, and RXFFINTCLR).
  - Release transmit and receive FIFO resets (TXFIFO and RXFIFORESET).
  - Release SPI FIFO channels from reset (SPIRST).

Step 3. If interrupts are used:

- In non-FIFO mode, enable the receiver overrun and/or SPI interrupts (OVERRUNINTENA and SPIINTENA).
- In FIFO mode, set the transmit and receive interrupt levels (TXFFIL and RXFFIL) then enable the interrupts (TXFFIENA and RXFFIENA).

Step 4. Set SPISWRESET to 1 to release the SPI from the reset state.

---

**NOTE:** Do not change the SPI configuration when communication is in progress.

---

### 22.4.3 Configuring the SPI for High-Speed Mode

In order to achieve the maximum rated speeds, the following settings must be made. This example assumes that the device is operating at 100 MHz.

Set LSPCLK equal to SYSCLK:

```
ClkCfgRegs.LOSPCP.bit.LSPCLKDIV = 0;
```

Select the appropriate Pin Mux options in GPIO\_CTRL\_REGS.

During the SPI configuration procedure:

Set HS\_MODE to 1.

```
SpiaRegs.SPICCR.bit.HS_MODE = 0x1;
```

Set SPIBRR to 3.  $SPICLK = LSPCLK / (SPIBRR + 1) = 25\text{MHz}$

```
SpiaRegs.SPIBRR = 0x3;
```

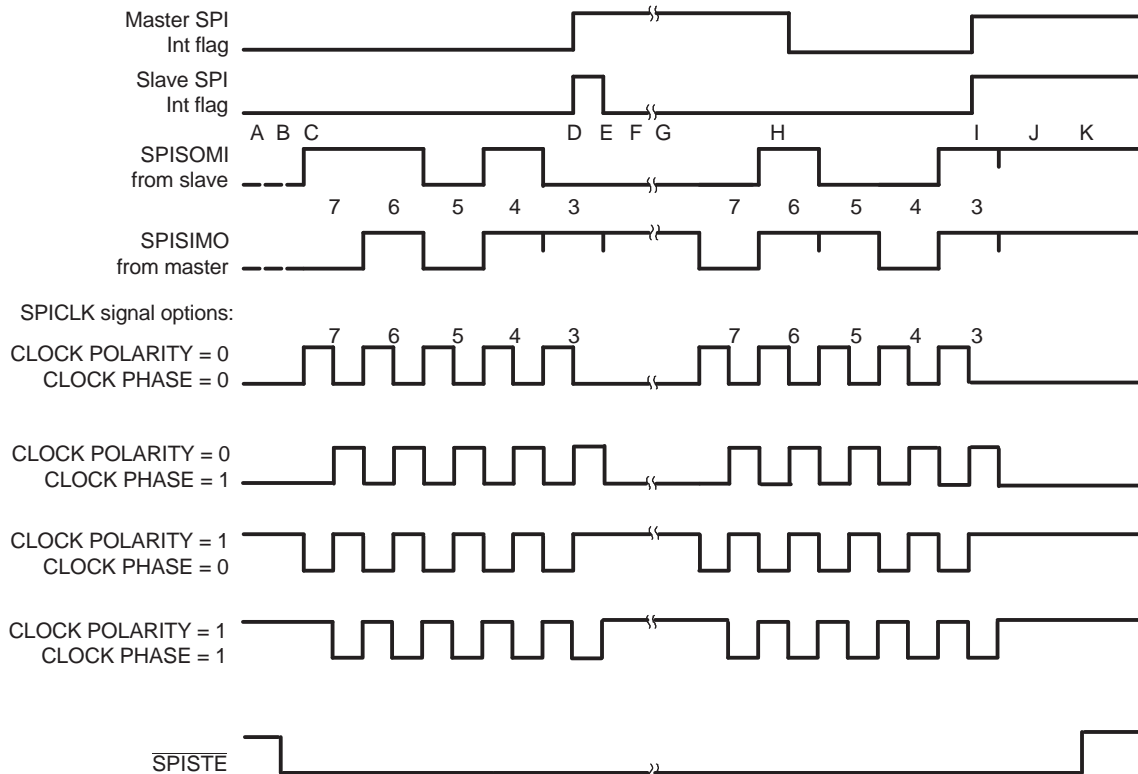
There are no other differences in the configuration from normal SPI operation. Sending and receiving data, interrupts, and DMA operation will operate without change.

### 22.4.4 Data Transfer Example

The timing diagram shown in [Figure 22-11](#) illustrates an SPI data transfer between two devices using a character length of five bits with the SPICLK being symmetrical.

The timing diagram with SPICLK asymmetrical ([Figure 22-8](#)) shares similar characterizations with [Figure 22-11](#) except that the data transfer is one LSPCLK cycle longer per bit during the low pulse (CLKPOLARITY = 0) or during the high pulse (CLKPOLARITY = 1) of the SPICLK.

[Figure 22-11](#) is applicable for 8-bit SPI only and is not for C28x devices that are capable of working with 16-bit data. The figure is shown for illustrative purposes only.

**Figure 22-11. Five Bits per Character**


- A Slave writes 0D0h to SPIDAT and waits for the master to shift out the data.
- B Master sets the slave  $\overline{\text{SPISTE}}$  signal low (active).
- C Master writes 058h to SPIDAT, which starts the transmission procedure.
- D First byte is finished and sets the interrupt flags.
- E Slave reads 0Bh from its SPIRXBUF (right-justified).
- F Slave writes 04Ch to SPIDAT and waits for the master to shift out the data.
- G Master writes 06Ch to SPIDAT, which starts the transmission procedure.
- H Master reads 01Ah from the SPIRXBUF (right-justified).
- I Second byte is finished and sets the interrupt flags.
- J Master reads 89h and the slave reads 8Dh from their respective SPIRXBUF. After the user's software masks off the unused bits, the master receives 09h and the slave receives 0Dh.
- K Master clears the slave  $\overline{\text{SPISTE}}$  signal high (inactive).

### 22.4.5 SPI 3-Wire Mode Code Examples

In addition to the normal SPI initialization, to configure the SPI module for 3-wire mode, the TRIWIRE bit (SPIPRI.0) must be set to 1. After initialization, there are several considerations to take into account when transmitting and receiving data in 3-wire master and slave mode. The following examples demonstrate these considerations.

In 3-wire master mode, SPICLKx, SPISTEx, and SPISIMOX pins must be configured as SPI pins (SPISOMIx pin can be configured as non-SPI pin). When the master transmits, it receives the data it transmits (because SPISIMOX and SPISOMIx are connected internally in 3-wire mode). Therefore, the junk data received must be cleared from the receive buffer every time data is transmitted.

#### Example 22-4. 3-Wire Master Mode Transmit

```

Uint16 data;
Uint16 dummy;
    
```



**Example 22-4. 3-Wire Master Mode Transmit (continued)**

```

SpiaRegs.SPICTL.bit.TALK = 1;           // Enable Transmit path
SpiaRegs.SPITXBUF = data; // Master transmits data
while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Waits until data rx'd
dummy = SpiaRegs.SPIRXBUF;             // Clears junk data from itself
                                         // bc it rx'd same data tx'd

```

To receive data in 3-wire master mode, the master must clear the TALK (SPICTL.1) bit to 0 to close the transmit path and then transmit dummy data in order to initiate the transfer from the slave. Because the TALK bit is 0, unlike in transmit mode, the master dummy data does not appear on the SPISIMOA pin, and the master does not receive its own dummy data. Instead, the data from the slave is received by the master.

**Example 22-5. 3-Wire Master Mode Receive**

```

Uint16 rdata;
Uint16 dummy;

SpiaRegs.SPICTL.bit.TALK = 0;           // Disable Transmit path
SpiaRegs.SPITXBUF = dummy;             // Send dummy to start tx
// NOTE: because TALK = 0, data does not tx onto SPISIMOA pin
while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Wait until data received
rdata = SpiaRegs.SPIRXBUF;             // Master reads data

```

In 3-wire slave mode, SPICLKx, SPISTEx, and SPISOMIx pins must be configured as SPI pins (SPISIMOA pin can be configured as non-SPI pin). Like in master mode, when transmitting, the slave receives the data it transmits and must clear this junk data from its receive buffer.

**Example 22-6. 3-Wire Slave Mode Transmit**

```

Uint16 data;
Uint16 dummy;
SpiaRegs.SPICTL.bit.TALK = 1;           // Enable Transmit path
SpiaRegs.SPITXBUF = data;             // Slave transmits data
while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Wait until data rx'd
dummy = SpiaRegs.SPIRXBUF;           // Clears junk data from itself

```

As in 3-wire master mode, the TALK bit must be cleared to 0. Otherwise, the slave receives data normally.

**Example 22-7. - 3-Wire Slave Mode Receive**

```

Uint16 rdata;
SpiaRegs.SPICTL.bit.TALK = 0;           // Disable Transmit path
while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Waits until data rx'd
rdata = SpiaRegs.SPIRXBUF;             // Slave reads data

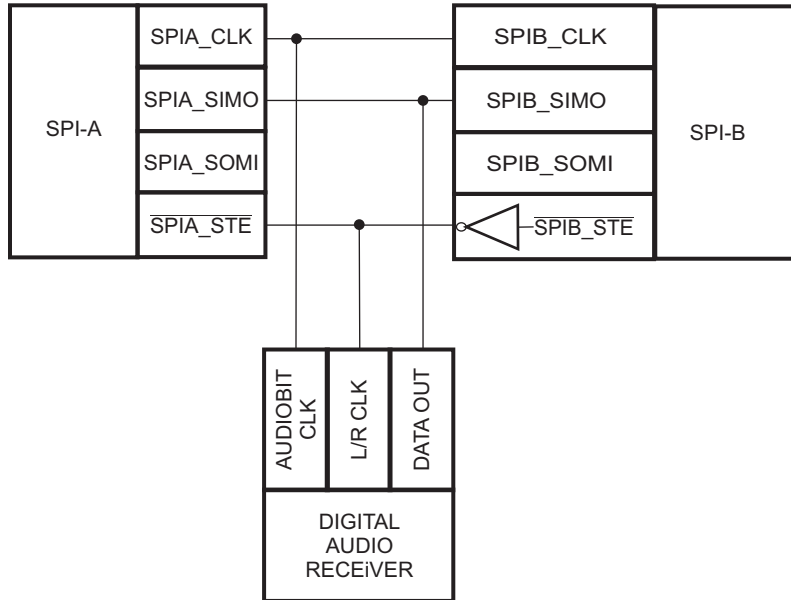
```

**22.4.6 SPI STEINV Bit in Digital Audio Transfers**

On those devices with two SPI modules, enabling the STEINV bit on one of the SPI modules allows the pair of SPIs to receive both left and right-channel digital audio data in slave mode. The SPI module that receives a normal active-low  $\overline{\text{SPISTE}}$  signal stores right-channel data, and the SPI module that receives an inverted active-high  $\overline{\text{SPISTE}}$  signal stores left-channel data from the master. To receive digital audio data from a digital audio interface receiver, the SPI modules can be connected as shown in [Figure 22-12](#).

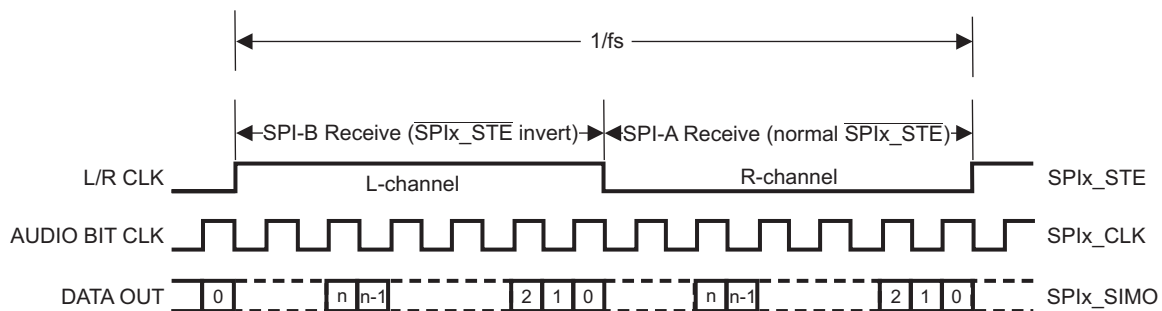
**NOTE:** This configuration is only applicable to slave mode (MASTER\_SLAVE = 0). When the SPI is configured as master (MASTER\_SLAVE = 1), the STEINV bit will have no effect on the SPISTE pin.

**Figure 22-12. SPI Digital Audio Receiver Configuration Using Two SPIs**



Standard C28x SPI timing requirements limit the number of digital audio interface formats supported using the 2-SPI configuration with the STEINV bit. See your device-specific data sheet electrical specifications for SPI timing requirements. With the SPI clock phase configured such that the CLKPOLARITY bit is 0 and the CLK\_PHASE bit is 1 (data latched on rising edge of clock), standard right-justified digital audio interface data format is supported as shown in Figure 22-13.

**Figure 22-13. Standard Right-Justified Digital Audio Data Format**



## 22.5 SPI Registers

This section describes the Serial Peripheral Interface registers. It is important to note that the SPI registers only allow 16-bit accesses.

### 22.5.1 SPI Base Addresses

**Table 22-6. SPI Base Address Table**

Device Registers	Register Name	Start Address	End Address
SpiaRegs	SPI_REGS	0x0000_6100	0x0000_610F
SpibRegs	SPI_REGS	0x0000_6110	0x0000_611F

## 22.5.2 SPI\_REGS Registers

Table 22-7 lists the SPI\_REGS registers. All register offset addresses not listed in Table 22-7 should be considered as reserved locations and the register contents should not be modified.

**Table 22-7. SPI\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	SPICCR	SPI Configuration Control Register		<a href="#">Go</a>
1h	SPICTL	SPI Operation Control Register		<a href="#">Go</a>
2h	SPISTS	SPI Status Register		<a href="#">Go</a>
4h	SPIBRR	SPI Baud Rate Register		<a href="#">Go</a>
6h	SPIRXEMU	SPI Emulation Buffer Register		<a href="#">Go</a>
7h	SPIRXBUF	SPI Serial Input Buffer Register		<a href="#">Go</a>
8h	SPITXBUF	SPI Serial Output Buffer Register		<a href="#">Go</a>
9h	SPIDAT	SPI Serial Data Register		<a href="#">Go</a>
Ah	SPIFFTX	SPI FIFO Transmit Register		<a href="#">Go</a>
Bh	SPIFFRX	SPI FIFO Receive Register		<a href="#">Go</a>
Ch	SPIFFCT	SPI FIFO Control Register		<a href="#">Go</a>
Fh	SPIPRI	SPI Priority Control Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 22-8 shows the codes that are used for access types in this section.

**Table 22-8. SPI\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
RC	R C	Read to Clear
<b>Write Type</b>		
W	W	Write
W1C	W 1C	Write 1 to clear
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 22.5.2.1 SPICCR Register (Offset = 0h) [reset = 0h]

SPICCR is shown in [Figure 22-14](#) and described in [Table 22-9](#).

Return to the [Summary Table](#).

SPICCR controls the setup of the SPI for operation.

**Figure 22-14. SPICCR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
SPISWRESET	CLKPOLARITY	HS_MODE	SPILBK	SPICCHAR			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			

**Table 22-9. SPICCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	SPISWRESET	R/W	0h	<p>SPI Software Reset</p> <p>When changing configuration, you should clear this bit before the changes and set this bit before resuming operation.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Initializes the SPI operating flags to the reset condition. Specifically, the RECEIVER OVERRUN Flag bit (SPISTS.7), the SPI INT FLAG bit (SPISTS.6), and the TXBUF FULL Flag bit (SPISTS.5) are cleared. SPISTE will become inactive. SPICLK will be immediately driven to 0 regardless of the clock polarity. The SPI configuration remains unchanged.</p> <p>1h (R/W) = SPI is ready to transmit or receive the next character. When the SPI SW RESET bit is a 0, a character written to the transmitter will not be shifted out when this bit is set. A new character must be written to the serial data register. SPICLK will be returned to its inactive state one SPICLK cycle after this bit is set.</p>
6	CLKPOLARITY	R/W	0h	<p>Shift Clock Polarity</p> <p>This bit controls the polarity of the SPICLK signal. CLOCK POLARITY and POLARITY CLOCK PHASE (SPICTL.3) control four clocking schemes on the SPICLK pin.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Data is output on rising edge and input on falling edge. When no SPI data is sent, SPICLK is at low level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows:</p> <ul style="list-style-type: none"> <li>- CLOCK PHASE = 0: Data is output on the rising edge of the SPICLK signal. Input data is latched on the falling edge of the SPICLK signal.</li> <li>- CLOCK PHASE = 1: Data is output one half-cycle before the first rising edge of the SPICLK signal and on subsequent falling edges of the SPICLK signal. Input data is latched on the rising edge of the SPICLK signal.</li> </ul> <p>1h (R/W) = Data is output on falling edge and input on rising edge. When no SPI data is sent, SPICLK is at high level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows:</p> <ul style="list-style-type: none"> <li>- CLOCK PHASE = 0: Data is output on the falling edge of the SPICLK signal. Input data is latched on the rising edge of the SPICLK signal.</li> <li>- CLOCK PHASE = 1: Data is output one half-cycle before the first falling edge of the SPICLK signal and on subsequent rising edges of the SPICLK signal. Input data is latched on the falling edge of the SPICLK signal.</li> </ul>

**Table 22-9. SPICCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	HS_MODE	R/W	0h	<p>High Speed Mode Enable Bits</p> <p>This bit determines if the High Speed mode is enabled. The correct GPIOs should be selected in the GPxGMUX/GPxMUX registers.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = SPI High Speed mode disabled. This is the default value after reset.</p> <p>1h (R/W) = SPI High Speed mode enabled,</p>
4	SPI_LBK	R/W	0h	<p>SPI Loopback Mode Select</p> <p>Loopback mode allows module validation during device testing. This mode is valid only in master mode of the SPI.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = SPI loopback mode disabled. This is the default value after reset.</p> <p>1h (R/W) = SPI loopback mode enabled, SIMO/SOMI lines are connected internally. Used for module self-tests.</p>
3-0	SPICHR	R/W	0h	<p>Character Length Control Bits</p> <p>These four bits determine the number of bits to be shifted in or SPI CHAR0 out as a single character during one shift sequence.</p> <p>SPICHR = Word length - 1</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = 1-bit word</p> <p>1h (R/W) = 2-bit word</p> <p>7h (R/W) = 8-bit word</p> <p>Fh (R/W) = 16-bit word</p>

### 22.5.2.2 SPICTL Register (Offset = 1h) [reset = 0h]

SPICTL is shown in [Figure 22-15](#) and described in [Table 22-10](#).

Return to the [Summary Table](#).

SPICTL controls data transmission, the SPI's ability to generate interrupts, the SPICLK phase, and the operational mode (slave or master).

**Figure 22-15. SPICTL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			OVERRUNINT ENA	CLK_PHASE	MASTER_SLA VE	TALK	SPIINTENA
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 22-10. SPICTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	OVERRUNINTENA	R/W	0h	<p>Overrun Interrupt Enable</p> <p>Overrun Interrupt Enable. Setting this bit causes an interrupt to be generated when the RECEIVER OVERRUN Flag bit (SPISTS.7) is set by hardware. Interrupts generated by the RECEIVER OVERRUN Flag bit and the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Disable RECEIVER OVERRUN interrupts.</p> <p>1h (R/W) = Enable RECEIVER_OVERRUN interrupts.</p>
3	CLK_PHASE	R/W	0h	<p>SPI Clock Phase Select</p> <p>This bit controls the phase of the SPICLK signal. CLOCK PHASE and CLOCK POLARITY (SPICCR.6) make four different clocking schemes possible (see clocking figures in SPI chapter). When operating with CLOCK PHASE high, the SPI (master or slave) makes the first bit of data available after SPIDAT is written and before the first edge of the SPICLK signal, regardless of which SPI mode is being used.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Normal SPI clocking scheme, depending on the CLOCK POLARITY bit (SPICCR.6).</p> <p>1h (R/W) = SPICLK signal delayed by one half-cycle. Polarity determined by the CLOCK POLARITY bit.</p>
2	MASTER_SLAVE	R/W	0h	<p>SPI Network Mode Control</p> <p>This bit determines whether the SPI is a network master or slave. SLAVE During reset initialization, the SPI is automatically configured as a network slave.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = SPI is configured as a slave.</p> <p>1h (R/W) = SPI is configured as a master.</p>

**Table 22-10. SPICTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	TALK	R/W	0h	<p>Transmit Enable</p> <p>The TALK bit can disable data transmission (master or slave) by placing the serial data output in the high-impedance state. If this bit is disabled during a transmission, the transmit shift register continues to operate until the previous character is shifted out. When the TALK bit is disabled, the SPI is still able to receive characters and update the status flags. TALK is cleared (disabled) by a system reset.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Disables transmission:</p> <ul style="list-style-type: none"> <li>- Slave mode operation: If not previously configured as a general-purpose I/O pin, the SPISOMI pin will be put in the high-impedance state.</li> <li>- Master mode operation: If not previously configured as a general-purpose I/O pin, the SPISIMO pin will be put in the high-impedance state.</li> </ul> <p>1h (R/W) = Enables transmission For the 4-pin option, ensure to enable the receiver's SPISTEn input pin.</p>
0	SPIINTENA	R/W	0h	<p>SPI Interrupt Enable</p> <p>This bit controls the SPI's ability to generate a transmit/receive interrupt. The SPI INT FLAG bit (SPISTS.6) is unaffected by this bit.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Disables the interrupt.</p> <p>1h (R/W) = Enables the interrupt.</p>



### 22.5.2.3 SPISTS Register (Offset = 2h) [reset = 0h]

SPISTS is shown in [Figure 22-16](#) and described in [Table 22-11](#).

Return to the [Summary Table](#).

SPISTS contains interrupt and status bits.

**Figure 22-16. SPISTS Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
OVERRUN_FL AG	INT_FLAG	BUFFULL_FL G	RESERVED				
W1C-0h	RC-0h	R-0h	R-0h				

**Table 22-11. SPISTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	OVERRUN_FLAG	W1C	0h	<p>SPI Receiver Overrun Flag</p> <p>This bit is a read/clear-only flag. The SPI hardware sets this bit when a receive or transmit operation completes before the previous character has been read from the buffer. The bit is cleared in one of three ways:</p> <ul style="list-style-type: none"> <li>- Writing a 1 to this bit</li> <li>- Writing a 0 to SPI SW RESET (SPICCR.7)</li> <li>- Resetting the system</li> </ul> <p>If the OVERRUN INT ENA bit (SPICTL.4) is set, the SPI requests only one interrupt upon the first occurrence of setting the RECEIVER OVERRUN Flag bit. Subsequent overruns will not request additional interrupts if this flag bit is already set. This means that in order to allow new overrun interrupt requests the user must clear this flag bit by writing a 1 to SPISTS.7 each time an overrun condition occurs. In other words, if the RECEIVER OVERRUN Flag bit is left set (not cleared) by the interrupt service routine, another overrun interrupt will not be immediately re-entered when the interrupt service routine is exited.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = A receive overrun condition has not occurred.</p> <p>1h (R/W) = The last received character has been overwritten and therefore lost (when the SPIRXBUF was overwritten by the SPI module before the previous character was read by the user application).</p> <p>Writing a '1' will clear this bit. The RECEIVER OVERRUN Flag bit should be cleared during the interrupt service routine because the RECEIVER OVERRUN Flag bit and SPI INT FLAG bit (SPISTS.6) share the same interrupt vector. This will alleviate any possible doubt as to the source of the interrupt when the next byte is received.</p>

**Table 22-11. SPISTS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	INT_FLAG	RC	0h	<p>SPI Interrupt Flag</p> <p>SPI INT FLAG is a read-only flag. Hardware sets this bit to indicate that the SPI has completed sending or receiving the last bit and is ready to be serviced. This flag causes an interrupt to be requested if the SPI INT ENA bit (SPICTL.0) is set. The received character is placed in the receiver buffer at the same time this bit is set. This bit is cleared in one of three ways:</p> <ul style="list-style-type: none"> <li>- Reading SPIRXBUF</li> <li>- Writing a 0 to SPI SW RESET (SPICCR.7)</li> <li>- Resetting the system</li> </ul> <p>Note: This bit should not be used if FIFO mode is enabled. The internal process of copying the received word from SPIRXBUF to the Receive FIFO will clear this bit. Use the FIFO status, or FIFO interrupt bits for similar functionality.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No full words have been received or transmitted. 1h (R/W) = Indicates that the SPI has completed sending or receiving the last bit and is ready to be serviced.</p>
5	BUFFULL_FLAG	R	0h	<p>SPI Transmit Buffer Full Flag</p> <p>This read-only bit gets set to 1 when a character is written to the SPI Transmit buffer SPITXBUF. It is cleared when the character is automatically loaded into SPIDAT when the shifting out of a previous character is complete.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Transmit buffer is not full. 1h (R/W) = Transmit buffer is full.</p>
4-0	RESERVED	R	0h	Reserved

### 22.5.2.4 SPIBRR Register (Offset = 4h) [reset = 0h]

SPIBRR is shown in [Figure 22-17](#) and described in [Table 22-12](#).

Return to the [Summary Table](#).

SPIBRR contains the bits used for baud-rate selection.

**Figure 22-17. SPIBRR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		SPI_BIT_RATE					
R-0h		R/W-0h					

**Table 22-12. SPIBRR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6-0	SPI_BIT_RATE	R/W	0h	<p>SPI Baud Rate Control</p> <p>These bits determine the bit transfer rate if the SPI is the network SPI BIT RATE 0 master. There are 125 data-transfer rates (each a function of the CPU clock, LSPCLK) that can be selected. One data bit is shifted per SPICLK cycle. (SPICLK is the baud rate clock output on the SPICLK pin.)</p> <p>If the SPI is a network slave, the module receives a clock on the SPICLK pin from the network master. Therefore, these bits have no effect on the SPICLK signal. The frequency of the input clock from the master should not exceed the slave SPI's LSPCLK signal divided by 4.</p> <p>In master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin. The SPI baud rates are determined by the following formula:</p> <p>For SPIBRR = 3 to 127: SPI Baud Rate = LSPCLK / (SPIBRR + 1)</p> <p>For SPIBRR = 0, 1, or 2: SPI Baud Rate = LSPCLK / 4</p> <p>Reset type: SYSRSn</p> <p>3h (R/W) = SPI Baud Rate = LSPCLK/4</p> <p>4h (R/W) = SPI Baud Rate = LSPCLK/5</p> <p>7Eh (R/W) = SPI Baud Rate = LSPCLK/127</p> <p>7Fh (R/W) = SPI Baud Rate = LSPCLK/128</p>

### 22.5.2.5 SPIRXEMU Register (Offset = 6h) [reset = 0h]

SPIRXEMU is shown in [Figure 22-18](#) and described in [Table 22-13](#).

Return to the [Summary Table](#).

SPIRXEMU contains the received data. Reading SPIRXEMU does not clear the SPI INT FLAG bit of SPISTS. This is not a real register but a dummy address from which the contents of SPIRXBUF can be read by the emulator without clearing the SPI INT FLAG.

**Figure 22-18. SPIRXEMU Register**

15	14	13	12	11	10	9	8
ERXBn							
R-0h							
7	6	5	4	3	2	1	0
ERXBn							
R-0h							

**Table 22-13. SPIRXEMU Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	ERXBn	R	0h	<p>Emulation Buffer Received Data</p> <p>SPIRXEMU functions almost identically to SPIRXBUF, except that reading SPIRXEMU does not clear the SPI INT FLAG bit (SPISTS.6). Once the SPIDAT has received the complete character, the character is transferred to SPIRXEMU and SPIRXBUF, where it can be read. At the same time, SPI INT FLAG is set.</p> <p>This mirror register was created to support emulation. Reading SPIRXBUF clears the SPI INT FLAG bit (SPISTS.6). In the normal operation of the emulator, the control registers are read to continually update the contents of these registers on the display screen. SPIRXEMU was created so that the emulator can read this register and properly update the contents on the display screen. Reading SPIRXEMU does not clear the SPI INT FLAG bit, but reading SPIRXBUF clears this flag. In other words, SPIRXEMU enables the emulator to emulate the true operation of the SPI more accurately.</p> <p>It is recommended that you view SPIRXEMU in the normal emulator run mode.</p> <p>Reset type: SYSRSn</p>

**22.5.2.6 SPIRXBUF Register (Offset = 7h) [reset = 0h]**

SPIRXBUF is shown in [Figure 22-19](#) and described in [Table 22-14](#).

Return to the [Summary Table](#).

SPIRXBUF contains the received data. Reading SPIRXBUF clears the SPI INT FLAG bit in SPISTS. If FIFO mode is enabled, reading this register will also decrement the RXFFST counter in SPIFFRX.

**Figure 22-19. SPIRXBUF Register**

15	14	13	12	11	10	9	8
RXBn							
R-0h							
7	6	5	4	3	2	1	0
RXBn							
R-0h							

**Table 22-14. SPIRXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RXBn	R	0h	<p>Received Data</p> <p>Once SPIDAT has received the complete character, the character is transferred to SPIRXBUF, where it can be read. At the same time, the SPI INT FLAG bit (SPISTS.6) is set. Since data is shifted into the SPI's most significant bit first, it is stored right-justified in this register.</p> <p>Reset type: SYSRSn</p>

### 22.5.2.7 SPITXBUF Register (Offset = 8h) [reset = 0h]

SPITXBUF is shown in [Figure 22-20](#) and described in [Table 22-15](#).

Return to the [Summary Table](#).

SPITXBUF stores the next character to be transmitted. Writing to this register sets the TX BUF FULL Flag bit in SPISTS. When the transmission of the current character is complete, the contents of this register are automatically loaded in SPIDAT and the TX BUF FULL Flag is cleared. If no transmission is currently active, data written to this register falls through into the SPIDAT register and the TX BUF FULL Flag is not set.

In master mode, if no transmission is currently active, writing to this register initiates a transmission in the same manner that writing to SPIDAT does.

**Figure 22-20. SPITXBUF Register**

15	14	13	12	11	10	9	8
TXBn							
R/W-0h							
7	6	5	4	3	2	1	0
TXBn							
R/W-0h							

**Table 22-15. SPITXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TXBn	R/W	0h	Transmit Data Buffer This is where the next character to be transmitted is stored. When the transmission of the current character has completed, if the TX BUF FULL Flag bit is set, the contents of this register is automatically transferred to SPIDAT, and the TX BUF FULL Flag is cleared. Writes to SPITXBUF must be left-justified. Reset type: SYSRSn

### 22.5.2.8 SPIDAT Register (Offset = 9h) [reset = 0h]

SPIDAT is shown in [Figure 22-21](#) and described in [Table 22-16](#).

Return to the [Summary Table](#).

SPIDAT is the transmit and receive shift register. Data written to SPIDAT is shifted out (MSB) on subsequent SPICLK cycles. For every bit (MSB) shifted out of the SPI, a bit is shifted into the LSB end of the shift register.

**Figure 22-21. SPIDAT Register**

15	14	13	12	11	10	9	8
SDATn							
R/W-0h							
7	6	5	4	3	2	1	0
SDATn							
R/W-0h							

**Table 22-16. SPIDAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SDATn	R/W	0h	<p>Serial Data Shift Register</p> <ul style="list-style-type: none"> <li>- It provides data to be output on the serial output pin if the TALK bit (SPICTL.1) is set.</li> <li>- When the SPI is operating as a master, a data transfer is initiated. When initiating a transfer, check the CLOCK POLARITY bit (SPICCR.6) described in Section 10.2.1.1 and the CLOCK PHASE bit (SPICTL.3) described in Section 10.2.1.2, for the requirements.</li> </ul> <p>In master mode, writing dummy data to SPIDAT initiates a receiver sequence. Since the data is not hardware-justified for characters shorter than sixteen bits, transmit data must be written in left-justified form, and received data read in right-justified form.</p> <p>Reset type: SYSRSn</p>

### 22.5.2.9 SPIFFTX Register (Offset = Ah) [reset = A000h]

SPIFFTX is shown in [Figure 22-22](#) and described in [Table 22-17](#).

Return to the [Summary Table](#).

SPIFFTX contains both control and status bits related to the output FIFO buffer. This includes FIFO reset control, FIFO interrupt level control, FIFO level status, as well as FIFO interrupt enable and clear bits.

**Figure 22-22. SPIFFTX Register**

15		14		13		12		11		10		9		8	
SPIRST		SPIFFENA		TXFIFO						TXFFST					
R/W-1h		R/W-0h		R/W-1h						R-0h					
7		6		5		4		3		2		1		0	
TXFFINT		TXFFINTCLR		TXFFIENA						TXFFIL					
R-0h		W-0h		R/W-0h						R/W-0h					

**Table 22-17. SPIFFTX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SPIRST	R/W	1h	SPI Reset Reset type: SYSRSn 0h (R/W) = Write 0 to reset the SPI transmit and receive channels. The SPI FIFO register configuration bits will be left as is. 1h (R/W) = SPI FIFO can resume transmit or receive. No effect to the SPI registers bits.
14	SPIFFENA	R/W	0h	SPI FIFO Enhancements Enable Reset type: SYSRSn 0h (R/W) = SPI FIFO enhancements are disabled. 1h (R/W) = SPI FIFO enhancements are enabled.
13	TXFIFO	R/W	1h	TX FIFO Reset Reset type: SYSRSn 0h (R/W) = Write 0 to reset the FIFO pointer to zero, and hold in reset. 1h (R/W) = Release transmit FIFO from reset.
12-8	TXFFST	R	0h	Transmit FIFO Status Reset type: SYSRSn 0h (R/W) = Transmit FIFO is empty. 1h (R/W) = Transmit FIFO has 1 word. 2h (R/W) = Transmit FIFO has 2 words. 10h (R/W) = Transmit FIFO has 16 words, which is the maximum. 1Fh (R/W) = Reserved.
7	TXFFINT	R	0h	TX FIFO Interrupt Flag Reset type: SYSRSn 0h (R/W) = TXFIFO interrupt has not occurred, This is a read-only bit. 1h (R/W) = TXFIFO interrupt has occurred, This is a read-only bit.
6	TXFFINTCLR	W	0h	TXFIFO Interrupt Clear Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on TXFFINT flag bit, Bit reads back a zero. 1h (R/W) = Write 1 to clear SPIFFTX[TXFFINT] flag.
5	TXFFIENA	R/W	0h	TX FIFO Interrupt Enable Reset type: SYSRSn 0h (R/W) = TX FIFO interrupt based on TXFFIL match (less than or equal to) will be disabled. 1h (R/W) = TX FIFO interrupt based on TXFFIL match (less than or equal to) will be enabled.



**Table 22-17. SPIFFTX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	TXFFIL	R/W	0h	<p>Transmit FIFO Interrupt Level Bits</p> <p>Transmit FIFO will generate interrupt when the FIFO status bits (TXFFST4-0) and FIFO level bits (TXFFIL4-0) match (less than or equal to).</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = A TX FIFO interrupt request is generated when there are no words remaining in the TX buffer.</p> <p>1h (R/W) = A TX FIFO interrupt request is generated when there is 1 word or no words remaining in the TX buffer.</p> <p>2h (R/W) = A TX FIFO interrupt request is generated when there is 2 words or fewer remaining in the TX buffer.</p> <p>10h (R/W) = A TX FIFO interrupt request is generated when there are 16 words or fewer remaining in the TX buffer.</p> <p>1Fh (R/W) = Reserved.</p>

**22.5.2.10 SPIFFRX Register (Offset = Bh) [reset = 201Fh]**

SPIFFRX is shown in [Figure 22-23](#) and described in [Table 22-18](#).

Return to the [Summary Table](#).

SPIFFRX contains both control and status bits related to the input FIFO buffer. This includes FIFO reset control, FIFO interrupt level control, FIFO level status, as well as FIFO interrupt enable and clear bits.

**Figure 22-23. SPIFFRX Register**

15	14	13	12	11	10	9	8
RXFFOVF	RXFFOVFCLR	RXFIFORESET	RXFFST				
R-0h	W-0h	R/W-1h	R-0h				
7	6	5	4	3	2	1	0
RXFFINT	RXFFINTCLR	RXFFIENA	RXFFIL				
R-0h	W-0h	R/W-0h	R/W-1Fh				

**Table 22-18. SPIFFRX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RXFFOVF	R	0h	Receive FIFO Overflow Flag Reset type: SYSRSn 0h (R/W) = Receive FIFO has not overflowed. This is a read-only bit. 1h (R/W) = Receive FIFO has overflowed, read-only bit. More than 16 words have been received in to the FIFO, and the first received word is lost.
14	RXFFOVFCLR	W	0h	Receive FIFO Overflow Clear Reset type: SYSRSn 0h (R/W) = Write 0 does not affect RXFFOVF flag bit, Bit reads back a zero. 1h (R/W) = Write 1 to clear SPIFFRX[RXFFOVF].
13	RXFIFORESET	R/W	1h	Receive FIFO Reset Reset type: SYSRSn 0h (R/W) = Write 0 to reset the FIFO pointer to zero, and hold in reset. 1h (R/W) = Re-enable receive FIFO operation.
12-8	RXFFST	R	0h	Receive FIFO Status Reset type: SYSRSn 0h (R/W) = Receive FIFO is empty. 1h (R/W) = Receive FIFO has 1 word. 2h (R/W) = Receive FIFO has 2 words. 10h (R/W) = Receive FIFO has 16 words, which is the maximum. 1Fh (R/W) = Reserved.
7	RXFFINT	R	0h	Receive FIFO Interrupt Flag Reset type: SYSRSn 0h (R/W) = RXFIFO interrupt has not occurred. This is a read-only bit. 1h (R/W) = RXFIFO interrupt has occurred. This is a read-only bit.
6	RXFFINTCLR	W	0h	Receive FIFO Interrupt Clear Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on RXFIFINT flag bit, Bit reads back a zero. 1h (R/W) = Write 1 to clear SPIFFRX[RXFFINT] flag

**Table 22-18. SPIFFRX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	RXFFIENA	R/W	0h	<p>RX FIFO Interrupt Enable</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = RX FIFO interrupt based on RXFFIL match (greater than or equal to) will be disabled.</p> <p>1h (R/W) = RX FIFO interrupt based on RXFFIL match (greater than or equal to) will be enabled.</p>
4-0	RXFFIL	R/W	1Fh	<p>Receive FIFO Interrupt Level Bits</p> <p>Receive FIFO generates an interrupt when the FIFO status bits (RXFFST4-0) are greater than or equal to the FIFO level bits (RXFFIL4-0). The default value of these bits after reset is 11111. This avoids frequent interrupts after reset, as the receive FIFO will be empty most of the time.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = A RX FIFO interrupt request is generated when there is 0 or more words in the RX buffer.</p> <p>1h (R/W) = A RX FIFO interrupt request is generated when there are 1 or more words in the RX buffer.</p> <p>2h (R/W) = A RX FIFO interrupt request is generated when there are 2 or more words in the RX buffer.</p> <p>10h (R/W) = A RX FIFO interrupt request is generated when there are 16 words in the RX buffer.</p> <p>1Fh (R/W) = Reserved.</p>

**22.5.2.11 SPIFFCT Register (Offset = Ch) [reset = 0h]**

SPIFFCT is shown in [Figure 22-24](#) and described in [Table 22-19](#).

Return to the [Summary Table](#).

SPIFFCT controls the FIFO transmit delay bits.

**Figure 22-24. SPIFFCT Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
TXDLY							
R/W-0h							

**Table 22-19. SPIFFCT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	TXDLY	R/W	0h	<p>FIFO Transmit Delay Bits</p> <p>These bits define the delay between every transfer from FIFO transmit buffer to transmit shift register. The delay is defined in number SPI serial clock cycles. The 8-bit register could define a minimum delay of 0 serial clock cycles and a maximum of 255 serial clock cycles. In FIFO mode, the buffer (TXBUF) between the shift register and the FIFO should be filled only after the shift register has completed shifting of the last bit. This is required to pass on the delay between transfers to the data stream. In the FIFO mode TXBUF should not be treated as one additional level of buffer.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The next word in the TX FIFO buffer is transferred to SPITXBUF immediately upon completion of transmission of the previous word.</p> <p>1h (R/W) = The next word in the TX FIFO buffer is transferred to SPITXBUF1 serial clock cycle after completion of transmission of the previous word.</p> <p>2h (R/W) = The next word in the TX FIFO buffer is transferred to SPITXBUF 2 serial clock cycles after completion of transmission of the previous word.</p> <p>FFh (R/W) = The next word in the TX FIFO buffer is transferred to SPITXBUF 255 serial clock cycles after completion of transmission of the previous word.</p>

**22.5.2.12 SPIPRI Register (Offset = Fh) [reset = 0h]**

SPIPRI is shown in [Figure 22-25](#) and described in [Table 22-20](#).

Return to the [Summary Table](#).

SPIPRI controls auxillary functions for the SPI including emulation control, SPISTE inversion, and 3-wire control.

**Figure 22-25. SPIPRI Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	RESERVED	SOFT	FREE	RESERVED		STEINV	TRIWIRES
R-0h		R/W-0h	R/W-0h	R-0h		R/W-0h	R/W-0h

**Table 22-20. SPIPRI Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	SOFT	R/W	0h	<p>Emulation Soft Run</p> <p>This bit only has an effect when the FREE bit is 0.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Transmission stops midway in the bit stream while TSUSPEND is asserted. Once TSUSPEND is deasserted without a system reset, the remainder of the bits pending in the DATBUF are shifted. Example: If SPIDAT has shifted 3 out of 8 bits, the communication freezes right there. However, if TSUSPEND is later deasserted without resetting the SPI, SPI starts transmitting from where it had stopped (fourth bit in this case) and will transmit 8 bits from that point.</p> <p>1h (R/W) = If the emulation suspend occurs before the start of a transmission, (that is, before the first SPICLK pulse) then the transmission will not occur. If the emulation suspend occurs after the start of a transmission, then the data will be shifted out to completion. When the start of transmission occurs is dependent on the baud rate used.</p> <p>Standard SPI mode: Stop after transmitting the words in the shift register and buffer. That is, after TXBUF and SPIDAT are empty.</p> <p>In FIFO mode: Stop after transmitting the words in the shift register and buffer. That is, after TX FIFO and SPIDAT are empty.</p>
4	FREE	R/W	0h	<p>Emulation Free Run</p> <p>These bits determine what occurs when an emulation suspend occurs (for example, when the debugger hits a breakpoint). The peripheral can continue whatever it is doing (free-run mode) or, if in stop mode, it can either stop immediately or stop when the current operation (the current receive/transmit sequence) is complete.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Emulation mode is selected by the SOFT bit</p> <p>1h (R/W) = Free run, continue SPI operation regardless of suspend or when the suspend occurred.</p>
3-2	RESERVED	R	0h	Reserved

**Table 22-20. SPIPRI Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	STEINV	R/W	0h	<p>SPISTEn Inversion Bit</p> <p>On devices with 2 SPI modules, inverting the SPISTE signal on one of the modules allows the device to receive left and right- channel digital audio data.</p> <p>This bit is only applicable to slave mode. Writing to this bit while configured as master (MASTER_SLAVE = 1) has no effect</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = SPISTEn is active low (normal)</p> <p>1h (R/W) = SPISTE is active high (inverted)</p>
0	TRIWIRE	R/W	0h	<p>SPI 3-wire Mode Enable</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Normal 4-wire SPI mode.</p> <p>1h (R/W) = 3-wire SPI mode enabled. The unused pin becomes a GPIO pin. In master mode, the SPISIMO pin becomes the SPIMOMI (master receive and transmit) pin and SPISOMI is free for non-SPI use. In slave mode, the SPISOMI pin becomes the SPISISO (slave receive and transmit) pin and SPISIMO is free for non-SPI use.</p>

### 22.5.3 Register to Driverlib Function Mapping

**Table 22-21. SPI Registers to Driverlib Functions**

File	Driverlib Function
<b>CCR</b>	
spi.c	SPI_setConfig
spi.c	SPI_clearInterruptStatus
spi.h	SPI_enableModule
spi.h	SPI_disableModule
spi.h	SPI_enableLoopback
spi.h	SPI_disableLoopback
spi.h	SPI_enableHighSpeedMode
spi.h	SPI_disableHighSpeedMode
<b>CTL</b>	
spi.c	SPI_setConfig
spi.c	SPI_enableInterrupt
spi.c	SPI_disableInterrupt
spi.h	SPI_enableTalk
spi.h	SPI_disableTalk
<b>STS</b>	
spi.c	SPI_getInterruptStatus
spi.c	SPI_clearInterruptStatus
spi.h	SPI_writeDataBlockingNonFIFO
spi.h	SPI_readDataBlockingNonFIFO
<b>BRR</b>	
spi.c	SPI_setConfig
spi.c	SPI_setBaudRate
<b>RXEMU</b>	
spi.h	SPI_readRxEmulationBuffer
<b>RXBUF</b>	
spi.h	SPI_readDataNonBlocking
spi.h	SPI_readDataBlockingFIFO
spi.h	SPI_readDataBlockingNonFIFO
<b>TXBUF</b>	
spi.h	SPI_writeDataNonBlocking
spi.h	SPI_writeDataBlockingFIFO
spi.h	SPI_writeDataBlockingNonFIFO
<b>FFTX</b>	
spi.c	SPI_enableInterrupt
spi.c	SPI_disableInterrupt
spi.c	SPI_getInterruptStatus
spi.c	SPI_clearInterruptStatus
spi.h	SPI_enableFIFO
spi.h	SPI_disableFIFO
spi.h	SPI_resetTxFIFO
spi.h	SPI_setFIFOInterruptLevel
spi.h	SPI_getFIFOInterruptLevel
spi.h	SPI_getTxFIFOStatus
spi.h	SPI_isBusy

**Table 22-21. SPI Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>FFRX</b>	
spi.c	SPI_enableInterrupt
spi.c	SPI_disableInterrupt
spi.c	SPI_getInterruptStatus
spi.c	SPI_clearInterruptStatus
spi.h	SPI_enableFIFO
spi.h	SPI_disableFIFO
spi.h	SPI_resetRxFIFO
spi.h	SPI_setFIFOInterruptLevel
spi.h	SPI_getFIFOInterruptLevel
spi.h	SPI_getRxFIFOStatus
<b>FFCT</b>	
spi.h	SPI_setTxFifoTransmitDelay
<b>PRI</b>	
spi.h	SPI_enableTriWire
spi.h	SPI_disableTriWire
spi.h	SPI_setSTESignalPolarity
spi.h	SPI_setEmulationMode



## Serial Communications Interface (SCI)

This chapter describes the features and operation of the serial communication interface (SCI) module. SCI is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format. The SCI receiver and transmitter each have a 16-level deep FIFO for reducing servicing overhead, and each has its own separate enable and interrupt bits. Both can be operated independently for half-duplex communication, or simultaneously for full-duplex communication.

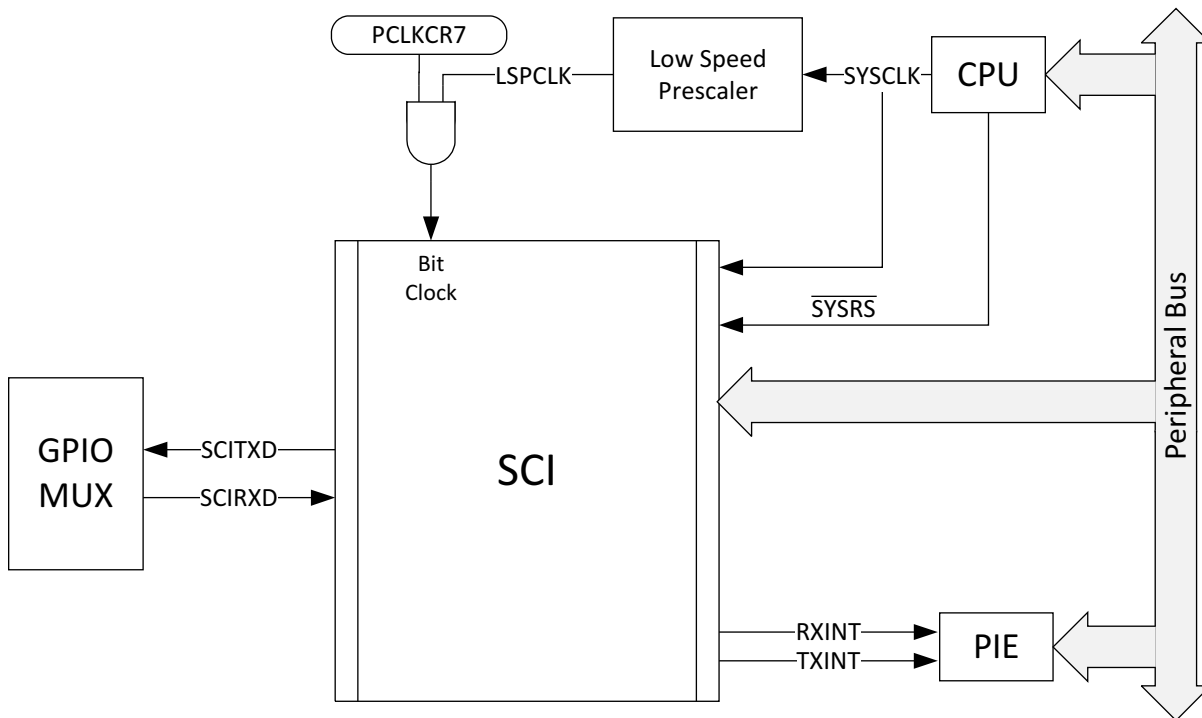
To specify data integrity, the SCI checks received data for break detection, parity, overrun, and framing errors. The bit rate is programmable to different speeds through a 16-bit baud-select register.

Topic	Page
23.1 Introduction .....	2250
23.2 Architecture .....	2252
23.3 SCI Module Signal Summary .....	2252
23.4 Configuring Device Pins .....	2252
23.5 Multiprocessor and Asynchronous Communication Modes .....	2252
23.6 SCI Programmable Data Format.....	2253
23.7 SCI Multiprocessor Communication .....	2253
23.8 Idle-Line Multiprocessor Mode .....	2254
23.9 Address-Bit Multiprocessor Mode.....	2256
23.10 SCI Communication Format .....	2257
23.11 SCI Port Interrupts .....	2259
23.12 SCI Baud Rate Calculations .....	2260
23.13 SCI Enhanced Features .....	2260
23.14 SCI Registers .....	2263

## 23.1 Introduction

The SCI interfaces are shown in [Figure 23-1](#).

**Figure 23-1. SCI CPU Interface**



Features of the SCI module include:

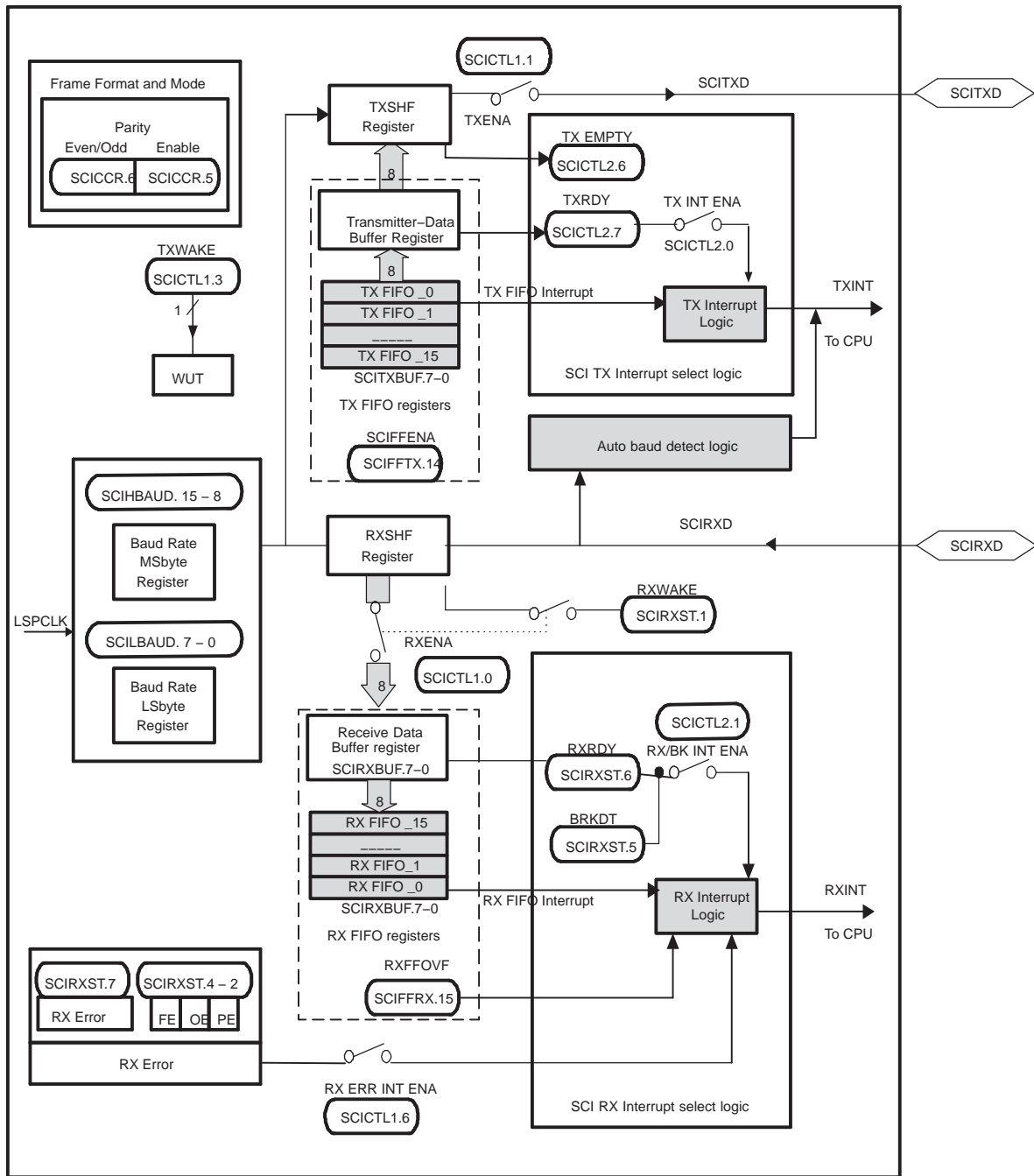
- Two external pins:
  - SCITXD: SCI transmit-output pin
  - SCIRXD: SCI receive-input pin
 Both pins can be used as GPIO if not used for SCI.
- Baud rate programmable to 64K different rates
- Data-word format
  - One start bit
  - Data-word length programmable from one to eight bits
  - Optional even/odd/no parity bit
  - One or two stop bits
  - An extra bit to distinguish addresses from data (address bit mode only)
- Four error-detection flags: parity, overrun, framing, and break detection
- Two wake-up multiprocessor modes: idle-line and address bit
- Half- or full-duplex operation
- Double-buffered receive and transmit functions
- Transmitter and receiver operations can be accomplished through interrupt-driven or polled algorithms with status flags.
- Separate enable bits for transmitter and receiver interrupts (except BRKDT)
- NRZ (non-return-to-zero) format

Enhanced features include:

- Auto-baud-detect hardware logic
- 16-level transmit/receive FIFO

Figure 23-2 shows the SCI module block diagram. The SCI port operation is configured and controlled by the registers listed in Section 23.14 of this chapter.

Figure 23-2. Serial Communications Interface (SCI) Module Block Diagram



## 23.2 Architecture

The major elements used in full-duplex operation are shown in [Figure 23-2](#) and include:

- A transmitter (TX) and its major registers (upper half of [Figure 23-2](#))
  - SCITXBUF — transmitter data buffer register. Contains data (loaded by the CPU) to be transmitted
  - TXSHF register — transmitter shift register. Accepts data from register SCITXBUF and shifts data onto the SCITXD pin, one bit at a time
- A receiver (RX) and its major registers (lower half of [Figure 23-2](#))
  - RXSHF register — receiver shift register. Shifts data in from SCIRXD pin, one bit at a time
  - SCIRXBUF — receiver data buffer register. Contains data to be read by the CPU. Data from a remote processor is loaded into register RXSHF and then into registers SCIRXBUF and SCIRXEMU
- A programmable baud generator
- Control and status registers

The SCI receiver and transmitter can operate either independently or simultaneously.

## 23.3 SCI Module Signal Summary

A summarized description of each SCI signal name is shown in [Table 23-1](#).

**Table 23-1. SCI Module Signal Summary**

Signal Name	Description
<b>External signals</b>	
SCIRXD	SCI Asynchronous Serial Port receive data
SCITXD	SCI Asynchronous Serial Port transmit data
<b>Control</b>	
Baud clock	LSPCLK Prescaled clock
<b>Interrupt signals</b>	
TXINT	Transmit interrupt
RXINT	Receive Interrupt

## 23.4 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

## 23.5 Multiprocessor and Asynchronous Communication Modes

The SCI has two multiprocessor protocols, the idle-line multiprocessor mode (see [Section 23.8](#)) and the address-bit multiprocessor mode (see [Section 23.9](#)). These protocols allow efficient data transfer between multiple processors.

The SCI offers the universal asynchronous receiver/transmitter (UART) communications mode for interfacing with many popular peripherals. The asynchronous mode (see [Section 23.10](#)) requires two lines to interface with many standard devices such as terminals and printers that use RS-232-C formats. Data transmission characteristics include:

- One start bit
- One to eight data bits

- An even/odd parity bit or no parity bit
- One or two stop bits

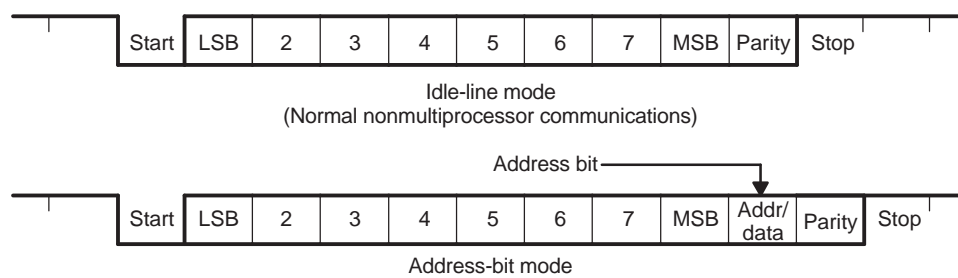
### 23.6 SCI Programmable Data Format

SCI data, both receive and transmit, is in NRZ (non-return-to-zero) format. The NRZ data format, shown in Figure 23-3, consists of:

- One start bit
- One to eight data bits
- An even/odd parity bit (optional)
- One or two stop bits
- An extra bit to distinguish addresses from data (address-bit mode only)

The basic unit of data is called a character and is one to eight bits in length. Each character of data is formatted with a start bit, one or two stop bits, and optional parity and address bits. A character of data with its formatting information is called a frame and is shown in Figure 23-3.

**Figure 23-3. Typical SCI Data Frame Formats**



To program the data format, use the SCICCR register. The bits used to program the data format are shown in Table 23-2.

**Table 23-2. Programming the Data Format Using SCICCR**

Bit(s)	Bit Name	Designation	Functions
2-0	SCI CHAR2-0	SCICCR.2:0	Select the character (data) length (one to eight bits).
5	PARITY ENABLE	SCICCR.5	Enables the parity function if set to 1, or disables the parity function if cleared to 0.
6	EVEN/ODD PARITY	SCICCR.6	If parity is enabled, selects odd parity if cleared to 0 or even parity if set to 1.
7	STOP BITS	SCICCR.7	Determines the number of stop bits transmitted—one stop bit if cleared to 0 or two stop bits if set to 1.

### 23.7 SCI Multiprocessor Communication

The multiprocessor communication format allows one processor to efficiently send blocks of data to other processors on the same serial link. On one serial line, there should be only one transfer at a time. In other words, there can be only one talker on a serial line at a time.

#### Address Byte

The first byte of a block of information that the talker sends contains an address byte that is read by all listeners. Only listeners with the correct address can be interrupted by the data bytes that follow the address byte. The listeners with an incorrect address remain uninterrupted until the next address byte.

### Sleep Bit

All processors on the serial link set the SCI SLEEP bit (bit 2 of SCICTL1) to 1 so that they are interrupted only when the address byte is detected. When a processor reads a block address that corresponds to the CPU device address as set by your application software, your program must clear the SLEEP bit to enable the SCI to generate an interrupt on receipt of each data byte.

Although the receiver still operates when the SLEEP bit is 1, it does not set RXRDY, RXINT, or any of the receiver error status bits to 1 unless the address byte is detected and the address bit in the received frame is a 1 (applicable to address-bit mode). The SCI does not alter the SLEEP bit; your software must alter the SLEEP bit.

#### 23.7.1 Recognizing the Address Byte

A processor recognizes an address byte differently, depending on the multiprocessor mode used. For example:

- The idle-line mode ([Section 23.8](#)) leaves a quiet space before the address byte. This mode does not have an extra address/data bit and is more efficient than the address-bit mode for handling blocks that contain more than ten bytes of data. The idle-line mode should be used for typical non-multiprocessor SCI communication.
- The address-bit mode ([Section 23.9](#)) adds an extra bit (that is, an address bit) into every byte to distinguish addresses from data. This mode is more efficient in handling many small blocks of data because, unlike the idle mode, it does not have to wait between blocks of data. However, at a high transmit speed, the program is not fast enough to avoid a 10-bit idle in the transmission stream.

#### 23.7.2 Controlling the SCI TX and RX Features

The multiprocessor mode is software selectable via the ADDR/IDLE MODE bit (SCICCR, bit 3). Both modes use the TXWAKE flag bit (SCICTL1, bit 3), RXWAKE flag bit (SCIRXST, bit1), and the SLEEP flag bit (SCICTL1, bit 2) to control the SCI transmitter and receiver features of these modes.

#### 23.7.3 Receipt Sequence

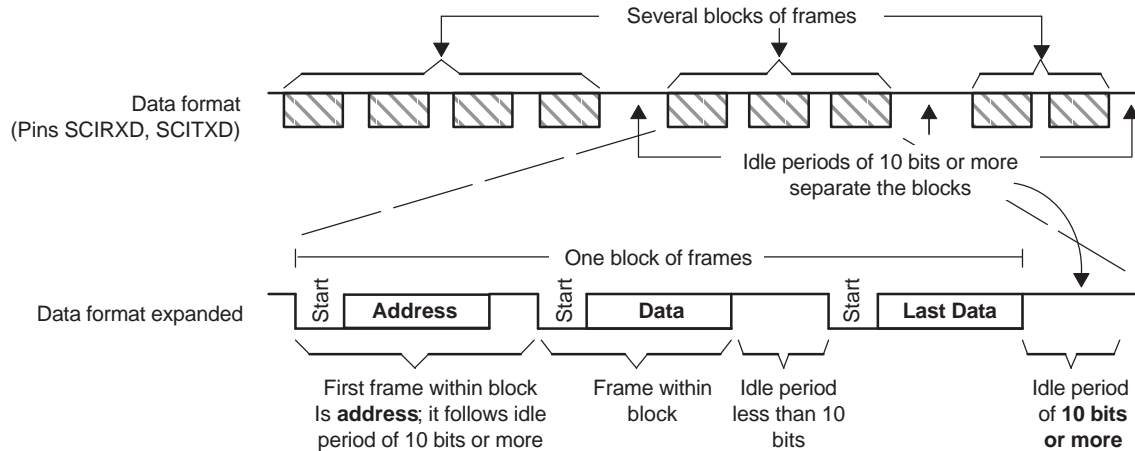
In both multiprocessor modes, the receive sequence is as follows:

1. At the receipt of an address block, the SCI port wakes up and requests an interrupt (bit number 1 RX/BK INT ENA-of SCICTL2 must be enabled to request an interrupt). It reads the first frame of the block, which contains the destination address.
2. A software routine is entered through the interrupt and checks the incoming address. This address byte is checked against its device address byte stored in memory.
3. If the check shows that the block is addressed to the device CPU, the CPU clears the SLEEP bit and reads the rest of the block. If not, the software routine exits with the SLEEP bit still set, and does not receive interrupts until the next block start.

### 23.8 Idle-Line Multiprocessor Mode

In the idle-line multiprocessor protocol (ADDR/IDLE MODE bit=0), blocks are separated by having a longer idle time between the blocks than between frames in the blocks. An idle time of ten or more high-level bits after a frame indicates the start of a new block. The time of a single bit is calculated directly from the baud value (bits per second). The idle-line multiprocessor communication format is shown in [Figure 23-4](#) (ADDR/IDLE MODE bit is bit 3 of SCICCR).

**Figure 23-4. Idle-Line Multiprocessor Communication Format**



### 23.8.1 Idle-Line Mode Steps

The steps followed by the idle-line mode:

- Step 1. SCI wakes up after receipt of the block-start signal.
- Step 2. The processor recognizes the next SCI interrupt.
- Step 3. The interrupt service routine compares the received address (sent by a remote transmitter) to its own.
- Step 4. If the CPU is being addressed, the service routine clears the SLEEP bit and receives the rest of the data block.
- Step 5. If the CPU is not being addressed, the SLEEP bit remains set. This lets the CPU continue to execute its main program without being interrupted by the SCI port until the next detection of a block start.

### 23.8.2 Block Start Signal

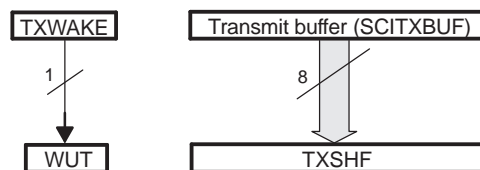
There are two ways to send a block-start signal:

1. **Method 1:** Deliberately leave an idle time of ten bits or more by delaying the time between the transmission of the last frame of data in the previous block and the transmission of the address frame of the new block.
2. **Method 2:** The SCI port first sets the TXWAKE bit (SCICTL1, bit 3) to 1 before writing to the SCITXBUF register. This sends an idle time of exactly 11 bits. In this method, the serial communications line is not idle any longer than necessary. (A don't care byte has to be written to SCITXBUF after setting TXWAKE, and before sending the address, so as to transmit the idle time.)

### 23.8.3 Wake-UP Temporary (WUT) Flag

Associated with the TXWAKE bit is the wake-up temporary (WUT) flag. WUT is an internal flag, double-buffered with TXWAKE. When TXSHF is loaded from SCITXBUF, WUT is loaded from TXWAKE, and the TXWAKE bit is cleared to 0. This arrangement is shown in [Figure 23-5](#).

**Figure 23-5. Double-Buffered WUT and TXSHF**



#### 23.8.3.1 Sending a Block Start Signal

To send out a block-start signal of exactly one frame time during a sequence of block transmissions:

1. Write a 1 to the TXWAKE bit.
2. Write a data word (content not important: a don't care) to the SCITXBUF register (transmit data buffer) to send a block-start signal. (The first data word written is suppressed while the block-start signal is sent out and ignored after that.) When the TXSHF (transmit shift register) is free again, SCITXBUF contents are shifted to TXSHF, the TXWAKE value is shifted to WUT, and then TXWAKE is cleared. Because TXWAKE was set to a 1, the start, data, and parity bits are replaced by an idle period of 11 bits transmitted following the last stop bit of the previous frame.
3. **Write a new address value to SCITXBUF**  
A don't-care data word must first be written to register SCITXBUF so that the TXWAKE bit value can be shifted to WUT. After the don't-care data word is shifted to the TXSHF register, the SCITXBUF (and TXWAKE if necessary) can be written to again because TXSHF and WUT are both double-buffered.

### 23.8.4 Receiver Operation

The receiver operates regardless of the SLEEP bit. However, the receiver neither sets RXRDY nor the error status bits, nor does it request a receive interrupt until an address frame is detected.

## 23.9 Address-Bit Multiprocessor Mode

In the address-bit protocol (ADDR/IDLE MODE bit=1), frames have an extra bit called an address bit that immediately follows the last data bit. The address bit is set to 1 in the first frame of the block and to 0 in all other frames. The idle period timing is irrelevant (see [Figure 23-6](#)).

### 23.9.1 Sending an Address

The TXWAKE bit value is placed in the address bit. During transmission, when the SCITXBUF register and TXWAKE are loaded into the TXSHF register and WUT respectively, TXWAKE is reset to 0 and WUT becomes the value of the address bit of the current frame. Thus, to send an address:

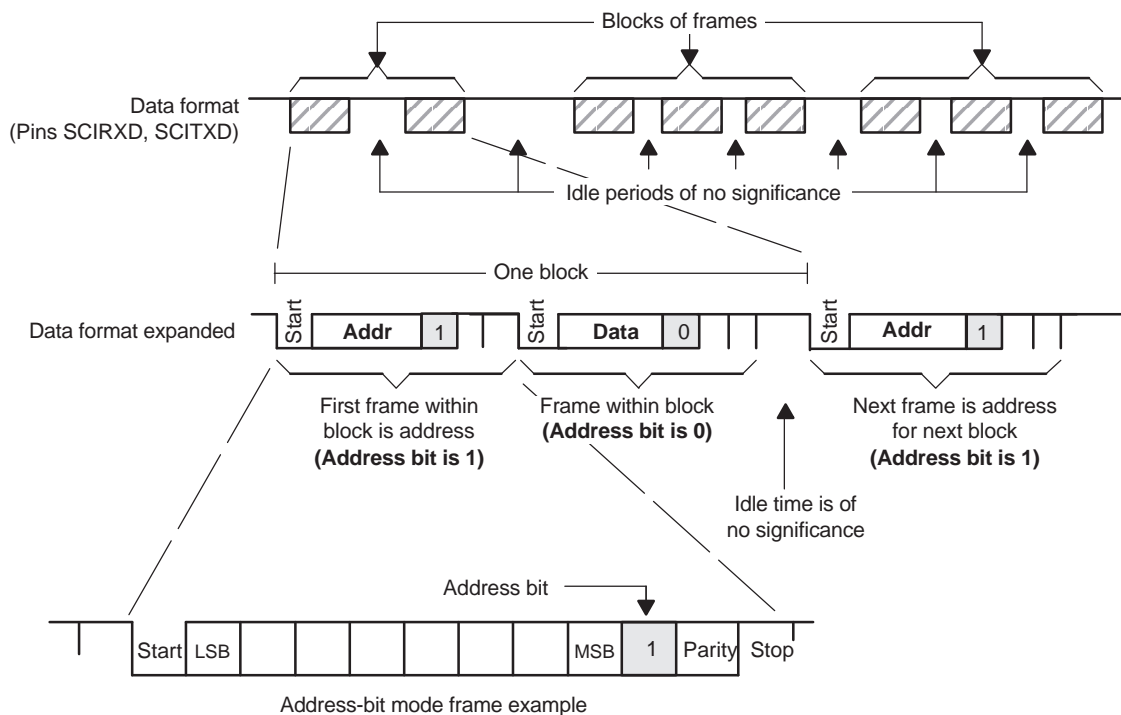
1. Set the TXWAKE bit to 1 and write the appropriate address value to the SCITXBUF register.



- When this address value is transferred to the TXSHF register and shifted out, its address bit is sent as a 1. This flags the other processors on the serial link to read the address.
- Write to SCITXBUF and TXWAKE after TXSHF and WUT are loaded. (Can be written to immediately since both TXSHF and WUT are both double-buffered.)
  - Leave the TXWAKE bit set to 0 to transmit non-address frames in the block.

**NOTE:** As a general rule, the address-bit format is typically used for data frames of 11 bytes or less. This format adds one bit value (1 for an address frame, 0 for a data frame) to all data bytes transmitted. The idle-line format is typically used for data frames of 12 bytes or more.

**Figure 23-6. Address-Bit Multiprocessor Communication Format**



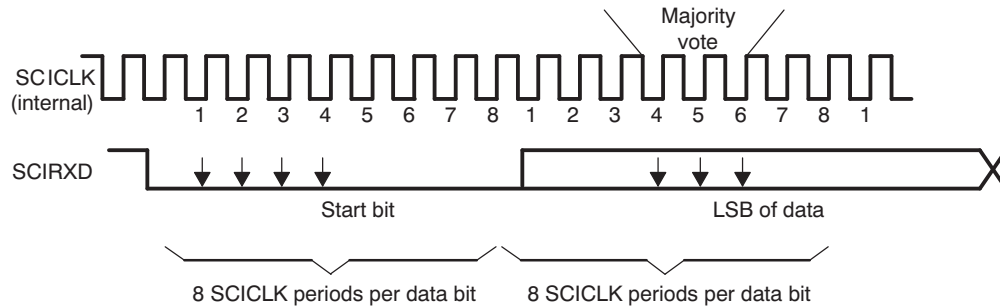
### 23.10 SCI Communication Format

The SCI asynchronous communication format uses either single line (one way) or two line (two way) communications. In this mode, the frame consists of a start bit, one to eight data bits, an optional even/odd parity bit, and one or two stop bits (shown in Figure 23-7). There are eight SCICLK periods per data bit.

The receiver begins operation on receipt of a valid start bit. A valid start bit is identified by four consecutive internal SCICLK periods of zero bits as shown in Figure 23-7. If any bit is not zero, then the processor starts over and begins looking for another start bit.

For the bits following the start bit, the processor determines the bit value by making three samples in the middle of the bits. These samples occur on the fourth, fifth, and sixth SCICLK periods, and bit-value determination is on a majority (two out of three) basis. Figure 23-7 illustrates the asynchronous communication format for this with a start bit showing where a majority vote is taken.

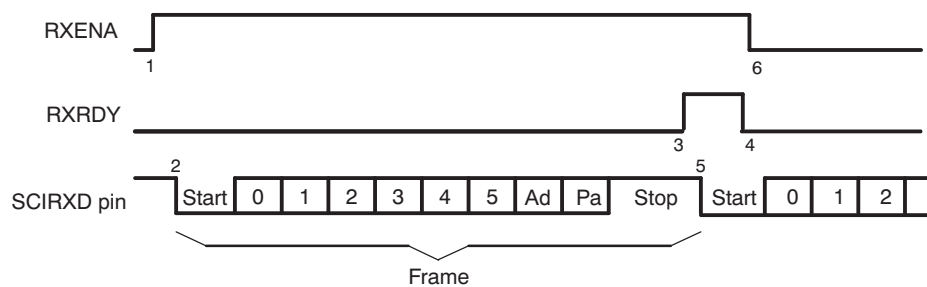
Since the receiver synchronizes itself to frames, the external transmitting and receiving devices do not have to use a synchronized serial clock. The clock can be generated locally.

**Figure 23-7. SCI Asynchronous Communications Format**


### 23.10.1 Receiver Signals in Communication Modes

Figure 23-8 illustrates an example of receiver signal timing that assumes the following conditions:

- Address-bit wake-up mode (address bit does not appear in idle-line mode)
- Six bits per character

**Figure 23-8. SCI RX Signals in Communication Modes**


- (1) Data arrives on the SCIRXD pin, start bit detected.
- (2) Bit RXENA is brought low to disable the receiver. Data continues to be assembled in RXSHF but is not transferred to the receiver buffer register.

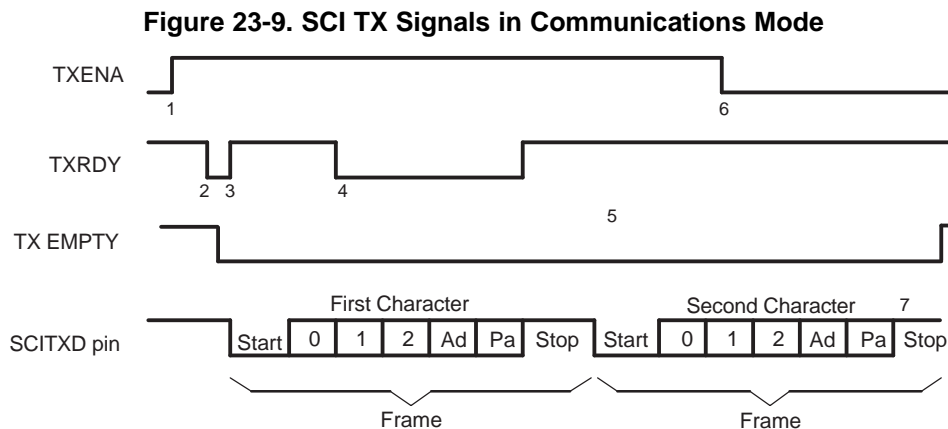
Notes:

1. Flag bit RXENA (SCICTL1, bit 0) goes high to enable the receiver.
2. Data arrives on the SCIRXD pin, start bit detected.
3. Data is shifted from RXSHF to the receiver buffer register (SCIRXBUF); an interrupt is requested. Flag bit RXRDY (SCIRXST, bit 6) goes high to signal that a new character has been received.
4. The program reads SCIRXBUF; flag RXRDY is automatically cleared.
5. The next byte of data arrives on the SCIRXD pin; the start bit is detected, then cleared.
6. Bit RXENA is brought low to disable the receiver. Data continues to be assembled in RXSHF but is not transferred to the receiver buffer register.

### 23.10.2 Transmitter Signals in Communication Modes

Figure 23-9 illustrates an example of transmitter signal timing that assumes the following conditions:

- Address-bit wake-up mode (address bit does not appear in idle-line mode)
- Three bits per character


**Notes:**

1. Bit TXENA (SCICTL1, bit 1) goes high, enabling the transmitter to send data.
2. SCITXBUF is written to; thus, (1) the transmitter is no longer empty, and (2) TXRDY goes low.
3. The SCI transfers data to the shift register (TXSHF). The transmitter is ready for a second character (TXRDY goes high), and it requests an interrupt (to enable an interrupt, bit TX INT ENA — SCICTL2, bit 0 — must be set).
4. The program writes a second character to SCITXBUF after TXRDY goes high (item 3). (TXRDY goes low again after the second character is written to SCITXBUF.)
5. Transmission of the first character is complete. Transfer of the second character to shift register TXSHF begins.
6. Bit TXENA goes low to disable the transmitter; the SCI finishes transmitting the current character.
7. Transmission of the second character is complete; transmitter is empty and ready for new character.

### 23.11 SCI Port Interrupts

The SCI receiver and transmitter can be interrupt controlled. The SCICTL2 register has one flag bit (TXRDY) that indicates active interrupt conditions, and the SCIRXST register has two interrupt flag bits (RXRDY and BRKDT), plus the RX ERROR interrupt flag which is a logical OR of the FE, OE, BRKDT, and PE conditions. The transmitter and receiver have separate interrupt-enable bits. When not enabled, the interrupts are not asserted; however, the condition flags remain active, reflecting transmission and receipt status.

The SCI has independent peripheral interrupt vectors for the receiver and transmitter. Peripheral interrupt requests can be either high priority or low priority. This is indicated by the priority bits which are output from the peripheral to the PIE controller. When both RX and TX interrupt requests are made at the same priority level, the receiver always has higher priority than the transmitter, reducing the possibility of receiver overrun.

The operation of peripheral interrupts is described in the peripheral interrupt expansion controller section of the *External Peripheral Interface (ePIE)* chapter.

- If the RX/BK INT ENA bit (SCICTL2, bit 1) is set, the receiver peripheral interrupt request is asserted when one of the following events occurs:
  - The SCI receives a complete frame and transfers the data in the RXSHF register to the SCIRXBUF register. This action sets the RXRDY flag (SCIRXST, bit 6) and initiates an interrupt.
  - A break detect condition occurs (the SCIRXD is low for ten bit periods following a missing stop bit). This action sets the BRKDT flag bit (SCIRXST, bit 5) and initiates an interrupt.
- If the TX INT ENA bit (SCICTL2.0) is set, the transmitter peripheral interrupt request is asserted whenever the data in the SCITXBUF register is transferred to the TXSHF register, indicating that the CPU can write to SCITXBUF; this action sets the TXRDY flag bit (SCICTL2, bit 7) and initiates an interrupt.

**NOTE:** Interrupt generation due to the RXRDY and BRKDT bits is controlled by the RX/BK INT ENA bit (SCICTL2, bit 1). Interrupt generation due to the RX ERROR bit is controlled by the RX ERR INT ENA bit (SCICTL1, bit 6).

## 23.12 SCI Baud Rate Calculations

The internally generated serial clock is determined by the low-speed peripheral clock (LSPCLK) and the baud-select registers. The SCI uses the 16-bit value of the baud-select registers to select one of the 64K different serial clock rates possible for a given LSPCLK.

See the bit descriptions in the baud-select registers, for the formula to use when calculating the SCI asynchronous baud. [Table 23-3](#) shows the baud-select values for common SCI bit rates.

**Table 23-3. Asynchronous Baud Register Values for Common SCI Bit Rates**

Ideal Baud	BRR	LSPCLK Clock Frequency, 100 MHz	
		Actual Baud	% Error
2400	5207 (1457h)	2400	0
4800	2603 (A2Bh)	4800	0
9600	1301 (515h)	9601	0.01
19200	650 (28Ah)	19201	0.01
38400	324 (144h)	38462	0.16

LSPCLK/16 is the maximum baud rate. For example, if LSPCLK is 100MHz, then the maximum baud rate is 6.25Mbps.

## 23.13 SCI Enhanced Features

The 28x SCI features autobaud detection and transmit/receive FIFO. The following section explains the FIFO operation.

### 23.13.1 SCI FIFO Description

The following steps explain the FIFO features and help with programming the SCI with FIFOs.

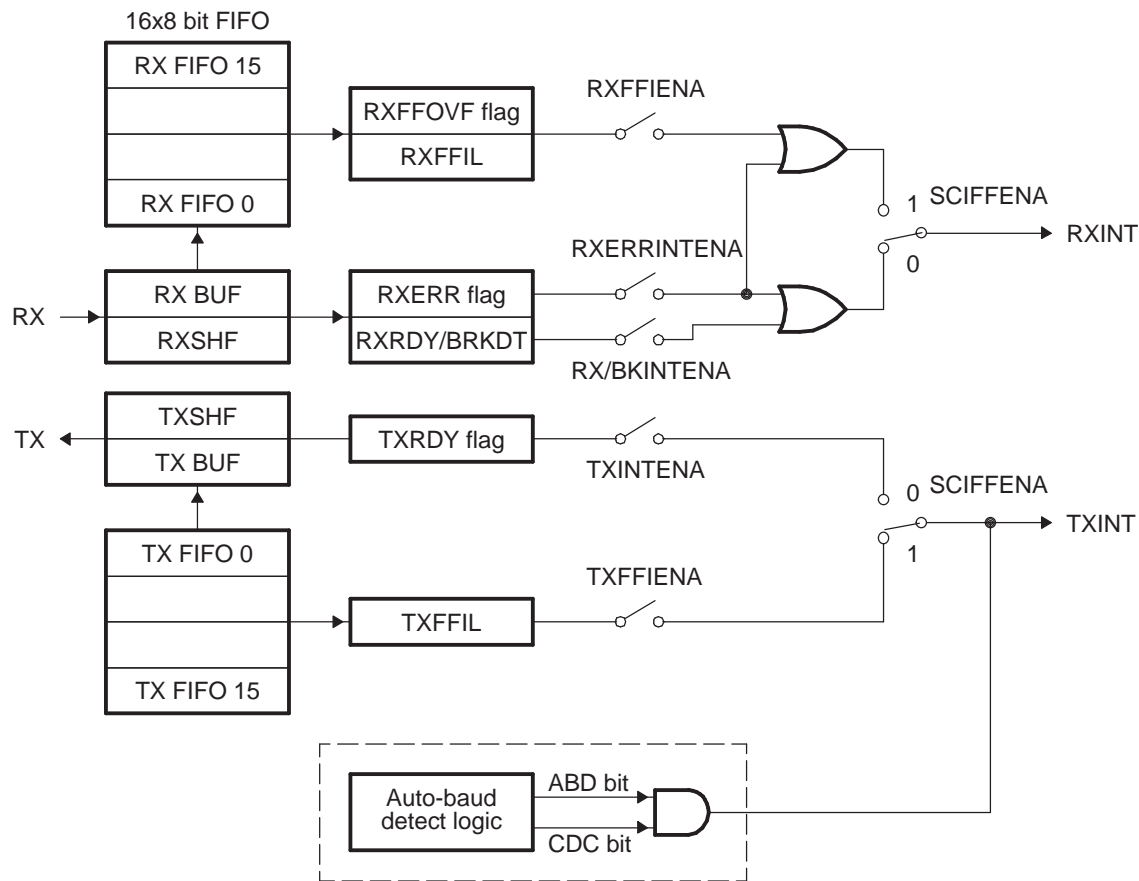
1. *Reset.* At reset the SCI powers up in standard SCI mode and the FIFO function is disabled. The FIFO registers SCIFFTX, SCIFFRX, and SCIFFCT remain inactive.
2. *Standard SCI.* The standard SCI modes will work normally with TXINT/RXINT interrupts as the interrupt source for the module.
3. *FIFO enable.* FIFO mode is enabled by setting the SCIFFEN bit in the SCIFFTX register. SCIRST can reset the FIFO mode at any stage of its operation.
4. *Active registers.* All the SCI registers and SCI FIFO registers (SCIFFTX, SCIFFRX, and SCIFFCT) are active.
5. *Interrupts.* FIFO mode has two interrupts; one for transmit FIFO, TXINT and one for receive FIFO, RXINT. RXINT is the common interrupt for SCI FIFO receive, receive error, and receive FIFO overflow conditions. The TXINT of the standard SCI will be disabled and this interrupt will service as SCI transmit FIFO interrupt.
6. *Buffers.* Transmit and receive buffers are supplemented with two 16-level FIFOs. The transmit FIFO registers are 8 bits wide and receive FIFO registers are 10 bits wide. The one-word transmit buffer of the standard SCI functions as a transition buffer between the transmit FIFO and shift register. The one-word transmit buffer is loaded from the transmit FIFO only after the last bit of the shift register is shifted out. With the FIFO enabled, TXSHF is directly loaded after an optional delay value (SCIFFCT), TXBUF is not used. When FIFO mode is enabled for SCI, characters written to SCITXBUF are queued in to SCI-TXFIFO and the characters received in SCI-RXFIFO can be read using SCIRXBUF.
7. *Delayed transfer.* The rate at which words in the FIFO are transferred to the transmit shift register is programmable. The SCIFFCT register bits (7–0) FFTXDLY7–FFTXDLY0 define the delay between the word transfer. The delay is defined in the number SCI baud clock cycles. The 8 bit register can define

a minimum delay of 0 baud clock cycles and a maximum of 256-baud clock cycles. With zero delay, the SCI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 256 clock delay the SCI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 256 baud clocks between each words. The programmable delay facilitates communication with slow SCI/UARTs with little CPU intervention.

8. *FIFO status bits.* Both the transmit and receive FIFOs have status bits TXFFST or RXFFST (bits 12–8) that define the number of words available in the FIFOs at any time. The transmit FIFO reset bit TXFIFO and receive reset bit RXFIFO reset the FIFO pointers to zero when these bits are cleared to 0. The FIFOs resumes operation from start once these bits are set to one.
9. *Programmable interrupt levels.* Both transmit and receive FIFO can generate CPU interrupts. The interrupt trigger is generated whenever the transmit FIFO status bits TXFFST (bits 12–8) match (less than or equal to) the interrupt trigger level bits TXFFIL (bits 4–0 ). This provides a programmable interrupt trigger for transmit and receive sections of the SCI. Default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO, respectively.

Figure 23-10 and Table 23-4 explain the operation/configuration of SCI interrupts in nonFIFO/FFO mode.

**Figure 23-10. SCI FIFO Interrupt Flags and Enable Logic**



**Table 23-4. SCI Interrupt Flags**

FIFO Options <sup>(1)</sup>	SCI Interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable SCIFFENA	Interrupt Line
SCI without FIFO	Receive error	RXERR <sup>(2)</sup>	RXERRINTENA	0	RXINT
	Receive break	BRKDT	RX/BKINTENA	0	RXINT
	Data receive	RXRDY	RX/BKINTENA	0	RXINT
	Transmit empty	TXRDY	TXINTENA	0	TXINT
SCI with FIFO	Receive error and receive break	RXERR	RXERRINTENA	1	RXINT
	FIFO receive	RXFFIL	RXFFIENA	1	RXINT
	Transmit empty	TXFFIL	TXFFIENA	1	TXINT
Auto-baud	Auto-baud detected	ABD	Don't care	x	TXINT

<sup>(1)</sup> FIFO mode TXSHF is directly loaded after delay value, TXBUF is not used.

<sup>(2)</sup> RXERR can be set by BRKDT, FE, OE, PE flags. In FIFO mode, BRKDT interrupt is only through RXERR flag

### 23.13.2 SCI Auto-Baud

Most SCI modules do not have an auto-baud detect logic built-in hardware. These SCI modules are integrated with embedded controllers whose clock rates are dependent on PLL reset values. Often embedded controller clocks change after final design. In the enhanced feature set this module supports an autobaud-detect logic in hardware. The following section explains the enabling sequence for autobaud-detect feature.

### 23.13.3 Autobaud-Detect Sequence

Bits ABD and CDC in SCIFFCT control the autobaud logic. The SCIRST bit should be enabled to make autobaud logic work.

If ABD is set while CDC is 1, which indicates auto-baud alignment, SCI transmit FIFO interrupt will occur (TXINT). After the interrupt service CDC bit has to be cleared by software. If CDC remains set even after interrupt service, there should be no repeat interrupts.

1. Enable autobaud-detect mode for the SCI by setting the CDC bit (bit 13) in SCIFFCT and clearing the ABD bit (Bit 15) by writing a 1 to ABDCLR bit (bit 14).
2. Initialize the baud register to be 1 or less than a baud rate limit of 500 Kbps.
3. Allow SCI to receive either character "A" or "a" from a host at the desired baud rate. If the first character is either "A" or "a", the autobaud- detect hardware will detect the incoming baud rate and set the ABD bit.
4. The auto-detect hardware will update the baud rate register with the equivalent baud value hex. The logic will also generate an interrupt to the CPU.
5. Respond to the interrupt clear ADB bit by writing a 1 to ABD CLR (bit 14) of SCIFFCT register and disable further autobaud locking by clearing CDC bit by writing a 0.
6. Read the receive buffer for character "A" or "a" to empty the buffer and buffer status.
7. If ABD is set while CDC is 1, which indicates autobaud alignment, the SCI transmit FIFO interrupt will occur (TXINT). After the interrupt service CDC bit must be cleared by software.

**NOTE:** At higher baud rates, the slew rate of the incoming data bits can be affected by transceiver and connector performance. While normal serial communications may work well, this slew rate may limit reliable autobaud detection at higher baud rates (typically beyond 100k baud) and cause the auto-baudlock feature to fail.

To avoid this, the following is recommended:

- Achieve a baud-lock between the host and 28x SCI boot loader using a lower baud rate.
- The host may then handshake with the loaded 28x application to set the SCI baud rate register to the desired higher baud rate.

## 23.14 SCI Registers

The section describes the Serial Communication Interface module Registers.

### 23.14.1 SCI Base Addresses

**Table 23-5. SCI Base Address Table**

Device Registers	Register Name	Start Address	End Address
SciaRegs	SCI_REGS	0x0000_7200	0x0000_720F
ScibRegs	SCI_REGS	0x0000_7210	0x0000_721F

### 23.14.2 SCI\_REGS Registers

Table 23-6 lists the SCI\_REGS registers. All register offset addresses not listed in Table 23-6 should be considered as reserved locations and the register contents should not be modified.

**Table 23-6. SCI\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	SCICCR	Communications control register		<a href="#">Go</a>
1h	SCICTL1	Control register 1		<a href="#">Go</a>
2h	SCIHBAUD	Baud rate (high) register		<a href="#">Go</a>
3h	SCILBAUD	Baud rate (low) register		<a href="#">Go</a>
4h	SCICTL2	Control register 2		<a href="#">Go</a>
5h	SCIRXST	Receive status register		<a href="#">Go</a>
6h	SCIRXEMU	Receive emulation buffer register		<a href="#">Go</a>
7h	SCIRXBUF	Receive data buffer		<a href="#">Go</a>
9h	SCITXBUF	Transmit data buffer		<a href="#">Go</a>
Ah	SCIFFTX	FIFO transmit register		<a href="#">Go</a>
Bh	SCIFFRX	FIFO receive register		<a href="#">Go</a>
Ch	SCIFFCT	FIFO control register		<a href="#">Go</a>
Fh	SCIPRI	SCI priority control		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 23-7 shows the codes that are used for access types in this section.

**Table 23-7. SCI\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R -0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1S	W 1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.



**23.14.2.1 SCICCR Register (Offset = 0h) [reset = 0h]**

SCICCR is shown in [Figure 23-11](#) and described in [Table 23-8](#).

Return to the [Summary Table](#).

SCICCR defines the character format, protocol, and communications mode used by the SCI.

**Figure 23-11. SCICCR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
STOPBITS	PARITY	PARITYENA	LOOPBKENA	ADDRIDLE_M ODE	SCICCHAR		
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		

**Table 23-8. SCICCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	STOPBITS	R/W	0h	SCI number of stop bits. This bit specifies the number of stop bits transmitted. The receiver checks for only one stop bit. Reset type: SYSRSn 0h (R/W) = One stop bit 1h (R/W) = Two stop bits
6	PARITY	R/W	0h	SCI parity odd/even selection. If the PARITY ENABLE bit (SCICCR, bit 5) is set, PARITY (bit 6) designates odd or even parity (odd or even number of bits with the value of 1 in both transmitted and received characters). Reset type: SYSRSn 0h (R/W) = Odd parity 1h (R/W) = Even parity
5	PARITYENA	R/W	0h	SCI parity enable. This bit enables or disables the parity function. If the SCI is in the addressbit multiprocessor mode (set using bit 3 of this register), the address bit is included in the parity calculation (if parity is enabled). For characters of less than eight bits, the remaining unused bits should be masked out of the parity calculation. Reset type: SYSRSn 0h (R/W) = Parity disabled no parity bit is generated during transmission or is expected during reception 1h (R/W) = Parity is enabled
4	LOOPBKENA	R/W	0h	Loop Back test mode enable. This bit enables the Loop Back test mode where the Tx pin is internally connected to the Rx pin. Reset type: SYSRSn 0h (R/W) = Loop Back test mode disabled 1h (R/W) = Loop Back test mode enabled

**Table 23-8. SCICCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	ADDRIDLE_MODE	R/W	0h	<p>SCI multiprocessor mode control bit.</p> <p>This bit selects one of the multiprocessor protocols. Multiprocessor communication is different from the other communication modes because it uses SLEEP and TXWAKE functions (bits SCICTL1, bit 2 and SCICTL1, bit 3, respectively). The idle-line mode is usually used for normal communications because the address-bit mode adds an extra bit to the frame. The idle-line mode does not add this extra bit and is compatible with RS-232 type communications.</p> <p>Reset type: SYSRSn  0h (R/W) = Idle-line mode protocol selected  1h (R/W) = Address-bit mode protocol selected</p>
2-0	SCICHAR	R/W	0h	<p>Character-length control bits 2-0.</p> <p>These bits select the SCI character length from one to eight bits. Characters of less than eight bits are right-justified in SCIRXBUF and SCIRXEMU and are padded with leading zeros in SCIRXBUF. SCITXBUF doesn't need to be padded with leading zeros.</p> <p>Reset type: SYSRSn  0h (R/W) = SCICHAR_LENGTH_1  1h (R/W) = SCICHAR_LENGTH_2  2h (R/W) = SCICHAR_LENGTH_3  3h (R/W) = SCICHAR_LENGTH_4  4h (R/W) = SCICHAR_LENGTH_5  5h (R/W) = SCICHAR_LENGTH_6  6h (R/W) = SCICHAR_LENGTH_7  7h (R/W) = SCICHAR_LENGTH_8</p>

### 23.14.2.2 SCICTL1 Register (Offset = 1h) [reset = 0h]

SCICTL1 is shown in [Figure 23-12](#) and described in [Table 23-9](#).

Return to the [Summary Table](#).

SCICTL1 controls the receiver/transmitter enable, TXWAKE and SLEEP functions, and the SCI software reset.

**Figure 23-12. SCICTL1 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	RXERRINTENA	SWRESET	RESERVED	TXWAKE	SLEEP	TXENA	RXENA
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 23-9. SCICTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	RXERRINTENA	R/W	0h	<p>SCI receive error interrupt enable.</p> <p>Setting this bit enables an interrupt if the RX ERROR bit (SCIRXST, bit 7) becomes set because of errors occurring.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Receive error interrupt disabled</p> <p>1h (R/W) = Receive error interrupt enabled</p>
5	SWRESET	R/W	0h	<p>SCI software reset (active low).</p> <p>Writing a 0 to this bit initializes the SCI state machines and operating flags (registers SCICTL2 and SCIRXST) to the reset condition. The SW RESET bit does not affect any of the configuration bits.</p> <p>All affected logic is held in the specified reset state until a 1 is written to SW RESET (the bit values following a reset are shown beneath each register diagram in this section). Thus, after a system reset, re-enable the SCI by writing a 1 to this bit. Clear this bit after a receiver break detect (BRKDT flag, bit SCIRXST, bit 5).</p> <p>SW RESET affects the operating flags of the SCI, but it neither affects the configuration bits nor restores the reset values. Once SW RESET is asserted, the flags are frozen until the bit is deasserted.</p> <p>The affected flags are as follows:</p> <p>Value After SW SCI Flag Register Bit</p> <p>RESET</p> <p>1 TXRDY SCICTL2, bit 7</p> <p>1 TX EMPTY SCICTL2, bit 6</p> <p>0 RXWAKE SCIRXST, bit 1</p> <p>0 PE SCIRXST, bit 2</p> <p>0 OE SCIRXST, bit 3</p> <p>0 FE SCIRXST, bit 4</p> <p>0 BRKDT SCIRXST, bit 5</p> <p>0 RXRDY SCIRXST, bit 6</p> <p>0 RX ERROR SCIRXST, bit 7</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Writing a 0 to this bit initializes the SCI state machines and operating flags (registers SCICTL2 and SCIRXST) to the reset condition.</p> <p>1h (R/W) = After a system reset, re-enable the SCI by writing a 1 to this bit.</p>

**Table 23-9. SCICTL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	RESERVED	R	0h	Reserved
3	TXWAKE	R/W	0h	<p>SCI transmitter wake-up method select.</p> <p>The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3)</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Transmit feature is not selected. In idle-line mode: write a 1 to TXWAKE, then write data to register SCITXBUF to generate an idle period of 11 data bits In address-bit mode: write a 1 to TXWAKE, then write data to SCITXBUF to set the address bit for that frame to 1</p> <p>1h (R/W) = Transmit feature selected is dependent on the mode, idle-line or address-bit: TXWAKE is not cleared by the SW RESET bit (SCICTL1, bit 5) it is cleared by a system reset or the transfer of TXWAKE to the WUT flag.</p>
2	SLEEP	R/W	0h	<p>SCI sleep.</p> <p>The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3). In a multiprocessor configuration, this bit controls the receiver sleep function. Clearing this bit brings the SCI out of the sleep mode.</p> <p>The receiver still operates when the SLEEP bit is set however, operation does not update the receiver buffer ready bit (SCIRXST, bit 6, RXRDY) or the error status bits (SCIRXST, bit 5-2: BRKDT, FE, OE, and PE) unless the address byte is detected. SLEEP is not cleared when the address byte is detected.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Sleep mode disabled</p> <p>1h (R/W) = Sleep mode enabled</p>
1	TXENA	R/W	0h	<p>SCI transmitter enable.</p> <p>Data is transmitted through the SCITXD pin only when TXENA is set. If reset, transmission is halted but only after all data previously written to SCITXBUF has been sent. Data written into SCITXBUF when TXENA is disabled will not be transmitted even if the TXENA is enabled later.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Transmitter disabled</p> <p>1h (R/W) = Transmitter enabled</p>
0	RXENA	R/W	0h	<p>SCI receiver enable.</p> <p>Data is received on the SCIRXD pin and is sent to the receiver shift register and then the receiver buffers. This bit enables or disables the receiver (transfer to the buffers).</p> <p>Clearing RXENA stops received characters from being transferred to the two receiver buffers and also stops the generation of receiver interrupts. However, the receiver shift register can continue to assemble characters. Thus, if RXENA is set during the reception of a character, the complete character will be transferred into the receiver buffer registers, SCIRXEMU and SCIRXBUF.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Prevent received characters from transfer into the SCIRXEMU and SCIRXBUF receiver buffers</p> <p>1h (R/W) = Send received characters to SCIRXEMU and SCIRXBUF</p>

**23.14.2.3 SCIHBAUD Register (Offset = 2h) [reset = 0h]**

SCIHBAUD is shown in [Figure 23-13](#) and described in [Table 23-10](#).

Return to the [Summary Table](#).

The values in SCIHBAUD and SCILBAUD specify the baud rate for the SCI.

**Figure 23-13. SCIHBAUD Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
BAUD							
R/W-0h							

**Table 23-10. SCIHBAUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	BAUD	R/W	0h	<p>SCI 16-bit baud selection Registers SCIHBAUD (MSbyte).</p> <p>The internally-generated serial clock is determined by the low speed peripheral clock (LSPCLK) signal and the two baud-select registers. The SCI uses the 16-bit value of these registers to select one of 64K serial clock rates for the communication modes.</p> <p><math>BRR = (SCIHBAUD \ll 8) + (SCILBAUD)</math></p> <p>The SCI baud rate is calculated using the following equation:</p> <p>SCI Asynchronous Baud = <math>LSPCLK / ((BRR + 1) * 8)</math></p> <p>Alternatively,</p> <p><math>BRR = LSPCLK / (SCI \text{ Asynchronous Baud} * 8) - 1</math></p> <p>Note that the above formulas are applicable only when <math>0 &lt; BRR &lt; 65536</math>. If <math>BRR = 0</math>, then</p> <p>SCI Asynchronous Baud = <math>LSPCLK / 16</math></p> <p>Where: BRR = the 16-bit value (in decimal) in the baud-select registers</p> <p>Reset type: SYSRSn</p>

**23.14.2.4 SCILBAUD Register (Offset = 3h) [reset = 0h]**

SCILBAUD is shown in [Figure 23-14](#) and described in [Table 23-11](#).

Return to the [Summary Table](#).

The values in SCIHBAUD and SCILBAUD specify the baud rate for the SCI.

**Figure 23-14. SCILBAUD Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
BAUD							
R/W-0h							

**Table 23-11. SCILBAUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	BAUD	R/W	0h	See SCIHBAUD Detailed Description Reset type: SYSRSn

### 23.14.2.5 SCICTL2 Register (Offset = 4h) [reset = C0h]

SCICTL2 is shown in [Figure 23-15](#) and described in [Table 23-12](#).

Return to the [Summary Table](#).

SCICTL2 enables the receive-ready, break-detect, and transmit-ready interrupts as well as transmitter-ready and -empty flags.

**Figure 23-15. SCICTL2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
TXRDY	TXEMPTY	RESERVED				RXBKINTENA	TXINTENA
R-1h	R-1h	R-0h				R/W-0h	R/W-0h

**Table 23-12. SCICTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	TXRDY	R	1h	Transmitter buffer register ready flag. When set, this bit indicates that the transmit data buffer register, SCITXBUF, is ready to receive another character. Writing data to the SCITXBUF automatically clears this bit. When set, this flag asserts a transmitter interrupt request if the interrupt-enable bit, TX INT ENA (SCICTL2.0), is also set. TXRDY is set to 1 by enabling the SW RESET bit (SCICTL1.5) or by a system reset. Reset type: SYSRSn 0h (R/W) = SCITXBUF is full 1h (R/W) = SCITXBUF is ready to receive the next character
6	TXEMPTY	R	1h	Transmitter empty flag. This flag's value indicates the contents of the transmitter's buffer register (SCITXBUF) and shift register (TXSHF). An active SW RESET (SCICTL1.5), or a system reset, sets this bit. This bit does not cause an interrupt request. Reset type: SYSRSn 0h (R/W) = Transmitter buffer or shift register or both are loaded with data 1h (R/W) = Transmitter buffer and shift registers are both empty
5-2	RESERVED	R	0h	Reserved
1	RXBKINTENA	R/W	0h	Receiver-buffer/break interrupt enable. This bit controls the interrupt request caused by either the RXRDY flag or the BRKDT flag (bits SCIRXST.6 and .5) being set. However, RX/BK INT ENA does not prevent the setting of these flags. Reset type: SYSRSn 0h (R/W) = Disable RXRDY/BRKDT interrupt 1h (R/W) = Enable RXRDY/BRKDT interrupt

**Table 23-12. SCICTL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	TXINTENA	R/W	0h	<p>SCITXBUF-register interrupt enable.</p> <p>This bit controls the interrupt request caused by the setting of TXRDY flag bit (SCICTL2.7). However, it does not prevent the TXRDY flag from being set (which indicates SCITXBUF is ready to receive another character).</p> <p>0 Disable TXRDY interrupt 1 Enable TXRDY interrupt.</p> <p>In non-FIFO mode, a dummy (or a valid) data has to be written to SCITXBUF for the first transmit interrupt to occur. This is the case when you enable the transmit interrupt for the first time and also when you re-enable (disable and then enable) the transmit interrupt. If TXINTENA is enabled after writing the data to SCITXBUF, it will not generate an interrupt.</p> <p>Reset type: SYSRSn 0h (R/W) = Disable TXRDY interrupt 1h (R/W) = Enable TXRDY interrupt</p>



### 23.14.2.6 SCIRXST Register (Offset = 5h) [reset = 0h]

SCIRXST is shown in [Figure 23-16](#) and described in [Table 23-13](#).

Return to the [Summary Table](#).

SCIRXST contains seven bits that are receiver status flags (two of which can generate interrupt requests). Each time a complete character is transferred to the receiver buffers (SCIRXEMU and SCIRXBUF), the status flags are updated.

**Figure 23-16. SCIRXST Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RXERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 23-13. SCIRXST Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	RXERROR	R	0h	<p>SCI receiver error flag.</p> <p>The RX ERROR flag indicates that one of the error flags in the receiver status register is set. RX ERROR is a logical OR of the break detect, framing error, overrun, and parity error enable flags (bits 5-2: BRKDT, FE, OE, and PE).</p> <p>A 1 on this bit will cause an interrupt if the RX ERR INT ENA bit (SCICTL1.6) is set. This bit can be used for fast error-condition checking during the interrupt service routine. This error flag cannot be cleared directly</p> <p>it is cleared by an active SW RESET or by a system reset.</p> <p>Reset type: SYSRSn 0h (R/W) = No error flags set 1h (R/W) = Error flag(s) set</p>
6	RXRDY	R	0h	<p>SCI receiver-ready flag.</p> <p>When a new character is ready to be read from the SCIRXBUF register, the receiver sets this bit, and a receiver interrupt is generated if the RX/BK INT ENA bit (SCICTL2.1) is a 1. RXRDY is cleared by a reading of the SCIRXBUF register, by an active SW RESET, or by a system reset.</p> <p>Reset type: SYSRSn 0h (R/W) = No new character in SCIRXBUF 1h (R/W) = Character ready to be read from SCIRXBUF</p>
5	BRKDT	R	0h	<p>SCI break-detect flag.</p> <p>The SCI sets this bit when a break condition occurs. A break condition occurs when the SCI receiver data line (SCIRXD) remains continuously low for at least ten bits, beginning after a missing first stop bit. The occurrence of a break causes a receiver interrupt to be generated if the RX/BK INT ENA bit is a 1, but it does not cause the receiver buffer to be loaded. A BRKDT interrupt can occur even if the receiver SLEEP bit is set to 1. BRKDT is cleared by an active SW RESET or by a system reset. It is not cleared by receipt of a character after the break is detected. In order to receive more characters, the SCI must be reset by toggling the SW RESET bit or by a system reset.</p> <p>Reset type: SYSRSn 0h (R/W) = No break condition 1h (R/W) = Break condition occurred</p>

**Table 23-13. SCIRXST Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	FE	R	0h	SCI framing-error flag. The SCI sets this bit when an expected stop bit is not found. Only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. The FE bit is reset by a clearing of the SW RESET bit or by a system reset. Reset type: SYSRSn 0h (R/W) = No framing error detected 1h (R/W) = Framing error detected
3	OE	R	0h	SCI overrun-error flag. The SCI sets this bit when a character is transferred into registers SCIRXEMU and SCIRXBUF before the previous character is fully read by the CPU or DMAC. The previous character is overwritten and lost. The OE flag bit is reset by an active SW RESET or by a system reset. Reset type: SYSRSn 0h (R/W) = No overrun error detected 1h (R/W) = Overrun error detected
2	PE	R	0h	SCI parity-error flag. This flag bit is set when a character is received with a mismatch between the number of 1s and its parity bit. The address bit is included in the calculation. If parity generation and detection is not enabled, the PE flag is disabled and read as 0. The PE bit is reset by an active SW RESET or a system reset. Reset type: SYSRSn 0h (R/W) = No parity error or parity is disabled 1h (R/W) = Parity error is detected
1	RXWAKE	R	0h	Receiver wake-up-detect flag Reset type: SYSRSn 0h (R/W) = No detection of a receiver wake-up condition 1h (R/W) = A value of 1 in this bit indicates detection of a receiver wake-up condition. In the address-bit multiprocessor mode (SCICCR.3 = 1), RXWAKE reflects the value of the address bit for the character contained in SCIRXBUF. In the idle-line multiprocessor mode, RXWAKE is set if the SCIRXD data line is detected as idle. RXWAKE is a read-only flag, cleared by one of the following: <ul style="list-style-type: none"> <li>- The transfer of the first byte after the address byte to SCIRXBUF (only in non-FIFO mode)</li> <li>- The reading of SCIRXBUF</li> <li>- An active SW RESET</li> <li>- A system reset</li> </ul>
0	RESERVED	R	0h	Reserved

### 23.14.2.7 SCIRXEMU Register (Offset = 6h) [reset = 0h]

SCIRXEMU is shown in [Figure 23-17](#) and described in [Table 23-14](#).

Return to the [Summary Table](#).

Normal SCI data-receive operations read the data received from the SCIRXBUF register. The SCIRXEMU register is used principally by the emulator (EMU) because it can continuously read the data received for screen updates without clearing the RXRDY flag. SCIRXEMU is cleared by a system reset. This is the register that should be used in an emulator watch window to view the contents of the SCIRXBUF register. SCIRXEMU is not physically implemented it is just a different address location to access the SCIRXBUF register without clearing the RXRDY flag.

**Figure 23-17. SCIRXEMU Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
ERXDT							
R-0h							

**Table 23-14. SCIRXEMU Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	ERXDT	R	0h	Receive emulation buffer data Reset type: SYSRSn

**23.14.2.8 SCIRXBUF Register (Offset = 7h) [reset = 0h]**

SCIRXBUF is shown in [Figure 23-18](#) and described in [Table 23-15](#).

Return to the [Summary Table](#).

When the current data received is shifted from RXSHF to the receiver buffer, flag bit RXRDY is set and the data is ready to be read. If the RXBKINTENA bit (SCICTL2.1) is set, this shift also causes an interrupt. When SCIRXBUF is read, the RXRDY flag is reset. SCIRXBUF is cleared by a system reset.

**Figure 23-18. SCIRXBUF Register**

15	14	13	12	11	10	9	8
SCIFFFE	SCIFFPE	RESERVED					
R-0h	R-0h	R-0h					
7	6	5	4	3	2	1	0
SAR							
R-0h							

**Table 23-15. SCIRXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SCIFFFE	R	0h	SCIFFFE. SCI FIFO Framing error flag bit (applicable only if the FIFO is enabled) Reset type: SYSRSn 0h (R/W) = No frame error occurred while receiving the character, in bits 7-0. This bit is associated with the character on the top of the FIFO. 1h (R/W) = A frame error occurred while receiving the character in bits 7-0. This bit is associated with the character on the top of the FIFO.
14	SCIFFPE	R	0h	SCIFFPE. SCI FIFO parity error flag bit (applicable only if the FIFO is enabled) Reset type: SYSRSn 0h (R/W) = No parity error occurred while receiving the character, in bits 7-0. This bit is associated with the character on the top of the FIFO. 1h (R/W) = A parity error occurred while receiving the character in bits 7-0. This bit is associated with the character on the top of the FIFO.
13-8	RESERVED	R	0h	Reserved
7-0	SAR	R	0h	Receive Character bits Reset type: SYSRSn

**23.14.2.9 SCITXBUF Register (Offset = 9h) [reset = 0h]**

SCITXBUF is shown in [Figure 23-19](#) and described in [Table 23-16](#).

Return to the [Summary Table](#).

Data bits to be transmitted are written to SCITXBUF. These bits must be rightjustified because the leftmost bits are ignored for characters less than eight bits long. The transfer of data from this register to the TXSHF transmitter shift register sets the TXRDY flag (SCICTL2.7), indicating that SCITXBUF is ready to receive another set of data. If bit TXINTENA (SCICTL2.0) is set, this data transfer also causes an interrupt.

**Figure 23-19. SCITXBUF Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
TXDT							
R/W-0h							

**Table 23-16. SCITXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	TXDT	R/W	0h	Transmit data buffer Reset type: SYSRSn

**23.14.2.10 SCIFFTX Register (Offset = Ah) [reset = A000h]**

SCIFFTX is shown in [Figure 23-20](#) and described in [Table 23-17](#).

Return to the [Summary Table](#).

SCIFFTX controls the transmit FIFO interrupt, FIFO enhancements, and reset for the SCI transmit and receive channels.

**Figure 23-20. SCIFFTX Register**

15		14		13		12		11		10		9		8	
SCIRST		SCIFFENA		TXFIFORESET						TXFFST					
R/W-1h		R/W-0h		R/W-1h						R-0h					
7		6		5		4		3		2		1		0	
TXFFINT		TXFFINTCLR		TXFFIENA						TXFFIL					
R-0h		R-0/W1S-0h		R/W-0h						R/W-0h					

**Table 23-17. SCIFFTX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SCIRST	R/W	1h	SCI Reset 0 Write 0 to reset the SCI transmit and receive channels. SCI FIFO register configuration bits will be left as is. 1 SCI FIFO can resume transmit or receive. SCIRST should be 1 even for Autobaud logic to work. Reset type: SYSRSn
14	SCIFFENA	R/W	0h	SCI FIFO enable Reset type: SYSRSn 0h (R/W) = SCI FIFO enhancements are disabled 1h (R/W) = SCI FIFO enhancements are enabled
13	TXFIFORESET	R/W	1h	Transmit FIFO reset Reset type: SYSRSn 0h (R/W) = Reset the FIFO pointer to zero and hold in reset 1h (R/W) = Re-enable transmit FIFO operation
12-8	TXFFST	R	0h	FIFO status Reset type: SYSRSn 0h (R/W) = Transmit FIFO is empty 1h (R/W) = Transmit FIFO has 1 words 2h (R/W) = Transmit FIFO has 2 words 3h (R/W) = Transmit FIFO has 3 words 4h (R/W) = Transmit FIFO has 4 words 5h (R/W) = Transmit FIFO has 5 words 6h (R/W) = Transmit FIFO has 6 words 7h (R/W) = Transmit FIFO has 7 words 8h (R/W) = Transmit FIFO has 8 words 9h (R/W) = Transmit FIFO has 9 words Ah (R/W) = Transmit FIFO has 10 words Bh (R/W) = Transmit FIFO has 11 words Ch (R/W) = Transmit FIFO has 12 words Dh (R/W) = Transmit FIFO has 13 words Eh (R/W) = Transmit FIFO has 14 words Fh (R/W) = Transmit FIFO has 15 words 10h (R/W) = Transmit FIFO has 16 words
7	TXFFINT	R	0h	Transmit FIFO interrupt Reset type: SYSRSn 0h (R/W) = TXFIFO interrupt has not occurred, read-only bit 1h (R/W) = TXFIFO interrupt has occurred, read-only bit

**Table 23-17. SCIFFTX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	TXFFINTCLR	R-0/W1S	0h	Transmit FIFO clear Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on TXFIFINT flag bit, Bit reads back a zero 1h (R/W) = Write 1 to clear TXFFINT flag in bit 7
5	TXFFIENA	R/W	0h	Transmit FIFO interrupt enable Reset type: SYSRSn 0h (R/W) = TX FIFO interrupt is disabled 1h (R/W) = TX FIFO interrupt is enabled. This interrupt is triggered whenever the transmit FIFO status (TXFFST) bits match (equal to or less than) the interrupt trigger level bits TXFFIL (bits 4-0).
4-0	TXFFIL	R/W	0h	TXFFIL4-0 Transmit FIFO interrupt level bits.  The transmit FIFO generates an interrupt whenever the FIFO status bits (TXFFST4-0) are less than or equal to the FIFO level bits (TXFFIL4-0). The maximum value that can be assigned to these bits to generate an interrupt cannot be more than the depth of the TX FIFO. The default value of these bits after reset is 00000b. Users should set TXFFIL to best fit their application needs by weighing between the CPU overhead to service the ISR and the best possible usage of SCI bus bandwidth.  Reset type: SYSRSn

**23.14.2.11 SCIFFRX Register (Offset = Bh) [reset = 201Fh]**

SCIFFRX is shown in [Figure 23-21](#) and described in [Table 23-18](#).

Return to the [Summary Table](#).

SCIFFTX controls the receive FIFO interrupt, receive FIFO reset, and status of the receive FIFO overflow.

**Figure 23-21. SCIFFRX Register**

15		14		13		12		11		10		9		8	
RXFFOVF		RXFFOVRCLR		RXFIFORESET						RXFFST					
R-0h		R-0/W1S-0h		R/W-1h						R-0h					
7		6		5		4		3		2		1		0	
RXFFINT		RXFFINTCLR		RXFFIENA						RXFFIL					
R-0h		W-0h		R/W-0h						R/W-1Fh					

**Table 23-18. SCIFFRX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RXFFOVF	R	0h	Receive FIFO overflow. This will function as flag, but cannot generate interrupt by itself. This condition will occur while receive interrupt is active. Receive interrupts should service this flag condition. Reset type: SYSRSn 0h (R/W) = Receive FIFO has not overflowed, read-only bit 1h (R/W) = Receive FIFO has overflowed, read-only bit. More than 16 words have been received in to the FIFO, and the first received word is lost
14	RXFFOVRCLR	R-0/W1S	0h	RXFFOVF clear Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on RXFFOVF flag bit, Bit reads back a zero 1h (R/W) = Write 1 to clear RXFFOVF flag in bit 15
13	RXFIFORESET	R/W	1h	Receive FIFO reset Reset type: SYSRSn 0h (R/W) = Write 0 to reset the FIFO pointer to zero, and hold in reset. 1h (R/W) = Re-enable receive FIFO operation
12-8	RXFFST	R	0h	FIFO status Reset type: SYSRSn 0h (R/W) = Receive FIFO is empty 1h (R/W) = Receive FIFO has 1 words 2h (R/W) = Receive FIFO has 2 words 3h (R/W) = Receive FIFO has 3 words 4h (R/W) = Receive FIFO has 4 words 5h (R/W) = Receive FIFO has 5 words 6h (R/W) = Receive FIFO has 6 words 7h (R/W) = Receive FIFO has 7 words 8h (R/W) = Receive FIFO has 8 words 9h (R/W) = Receive FIFO has 9 words Ah (R/W) = Receive FIFO has 10 words Bh (R/W) = Receive FIFO has 11 words Ch (R/W) = Receive FIFO has 12 words Dh (R/W) = Receive FIFO has 13 words Eh (R/W) = Receive FIFO has 14 words Fh (R/W) = Receive FIFO has 15 words 10h (R/W) = Receive FIFO has 16 words



**Table 23-18. SCIFFRX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	RXFFINT	R	0h	Receive FIFO interrupt Reset type: SYSRSn 0h (R/W) = RXFIFO interrupt has not occurred, read-only bit 1h (R/W) = RXFIFO interrupt has occurred, read-only bit
6	RXFFINTCLR	W	0h	Receive FIFO interrupt clear Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on RXFIFINT flag bit. Bit reads back a zero. 1h (R/W) = Write 1 to clear RXFFINT flag in bit 7
5	RXFFIENA	R/W	0h	Receive FIFO interrupt enable Reset type: SYSRSn 0h (R/W) = RX FIFO interrupt is disabled 1h (R/W) = RX FIFO interrupt is enabled. This interrupt is triggered whenever the receive FIFO status (RXFFST) bits match (equal to or greater than) the interrupt trigger level bits RXFFIL (bits 4-0).
4-0	RXFFIL	R/W	1Fh	Receive FIFO interrupt level bits  The receive FIFO generates an interrupt whenever the FIFO status bits (RXFFST4-0) are greater than or equal to the FIFO level bits (RXFFIL4-0). The maximum value that can be assigned to these bits to generate an interrupt cannot be more than the depth of the RX FIFO. The default value of these bits after reset is 11111b. Users should set RXFFIL to best fit their application needs by weighing between the CPU overhead to service the ISR and the best possible usage of received SCI data.  Reset type: SYSRSn

**23.14.2.12 SCIFFCT Register (Offset = Ch) [reset = 0h]**

SCIFFCT is shown in [Figure 23-22](#) and described in [Table 23-19](#).

Return to the [Summary Table](#).

SCIFFCT contains the status of auto-baud detect, clears the auto-baud flag, and calibrate for A-detect bit.

**Figure 23-22. SCIFFCT Register**

15		14		13		12		11		10		9		8	
ABD		ABDCLR		CDC		RESERVED									
R-0h		W-0h		R/W-0h		R-0h									
7		6		5		4		3		2		1		0	
FFTXDLY															
R/W-0h															

**Table 23-19. SCIFFCT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	ABD	R	0h	Auto-baud detect (ABD) bit Reset type: SYSRSn 0h (R/W) = Auto-baud detection is not complete. "A","a" character has not been received successfully. 1h (R/W) = Auto-baud hardware has detected "A" or "a" character on the SCI receive register. Auto-detect is complete.
14	ABDCLR	W	0h	ABD-clear bit Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on ABD flag bit. Bit reads back a zero. 1h (R/W) = Write 1 to clear ABD flag in bit 15.
13	CDC	R/W	0h	CDC calibrate A-detect bit Reset type: SYSRSn 0h (R/W) = Disables auto-baud alignment 1h (R/W) = Enables auto-baud alignment
12-8	RESERVED	R	0h	Reserved
7-0	FFTXDLY	R/W	0h	FIFO transfer delay. These bits define the delay between every transfer from FIFO transmit bufferto transmit shift register. The delay is defined in the number of SCI serial baud clock cycles. The 8 bit register could define a minimum delay of 0 baud clock cycles and a maximum of 256 baud clock cycles  In FIFO mode, the buffer (TXBUF) between the shift register and the FIFO should be filled only after the shift register has completed shifting of the last bit. This is required to pass on the delay between transfers to the data stream. In FIFO mode, TXBUF should not be treated as one additional level of buffer. The delayed transmit feature will help to create an auto-flow scheme without RTS/CTS controls as in standard UARTS.  When SCI is configured for one stop-bit, delay introduced by FFTXDLY between one frame and the next frame is equal to number of baud clock cycles that FFTXDLY is set to.  When SCI is configured for two stop-bits, delay introduced by FFTXDLY between one frame and the next frame is equal to number of baud clock cycles that FFTXDLY is set to minus 1.  Reset type: SYSRSn

**23.14.2.13 SCIPRI Register (Offset = Fh) [reset = 0h]**

SCIPRI is shown in [Figure 23-23](#) and described in [Table 23-20](#).

Return to the [Summary Table](#).

SCIPRI determines what happens when an emulation suspend event occurs.

**Figure 23-23. SCIPRI Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FREESOFT			RESERVED	
R-0h			R/W-0h			R-0h	

**Table 23-20. SCIPRI Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-5	RESERVED	R	0h	Reserved
4-3	FREESOFT	R/W	0h	<p>These bits determine what occurs when an emulation suspend event occurs (for example, when the debugger hits a breakpoint). The peripheral can continue whatever it is doing (free-run mode), or if in stop mode, it can either stop immediately or stop when the current operation (the current receive/transmit sequence) is complete.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Immediate stop on suspend</p> <p>1h (R/W) = Complete current receive/transmit sequence before stopping</p> <p>2h (R/W) = Free run</p> <p>3h (R/W) = Free run</p>
2-0	RESERVED	R	0h	Reserved

### 23.14.3 Register to Driverlib Function Mapping

**Table 23-21. SCI Registers to Driverlib Functions**

File	Driverlib Function
<b>CCR</b>	
sci.c	SCI_setConfig
sci.h	SCI_setParityMode
sci.h	SCI_getParityMode
sci.h	SCI_getConfig
sci.h	SCI_enableLoopback
sci.h	SCI_disableLoopback
<b>CTL1</b>	
sci.c	SCI_enableInterrupt
sci.c	SCI_disableInterrupt
sci.h	SCI_enableModule
sci.h	SCI_disableModule
sci.h	SCI_performSoftwareReset
<b>HBAUD</b>	
sci.c	SCI_setConfig
sci.h	SCI_lockAutobaud
sci.h	SCI_getConfig
<b>LBAUD</b>	
sci.c	SCI_setConfig
sci.h	SCI_lockAutobaud
sci.h	SCI_getConfig
<b>CTL2</b>	
sci.c	SCI_enableInterrupt
sci.c	SCI_disableInterrupt
sci.c	SCI_getInterruptStatus
sci.h	SCI_isSpaceAvailableNonFIFO
sci.h	SCI_isTransmitterBusy
<b>RXST</b>	
sci.c	SCI_getInterruptStatus
sci.h	SCI_isDataAvailableNonFIFO
sci.h	SCI_getRxStatus
<b>RXBUF</b>	
sci.c	SCI_readCharArray
sci.h	SCI_readCharBlockingFIFO
sci.h	SCI_readCharBlockingNonFIFO
sci.h	SCI_readCharNonBlocking
<b>TXBUF</b>	
sci.c	SCI_writeCharArray
sci.h	SCI_writeCharBlockingFIFO
sci.h	SCI_writeCharBlockingNonFIFO
sci.h	SCI_writeCharNonBlocking
<b>FFTX</b>	
sci.c	SCI_enableInterrupt
sci.c	SCI_disableInterrupt
sci.c	SCI_getInterruptStatus
sci.c	SCI_clearInterruptStatus

**Table 23-21. SCI Registers to Driverlib Functions (continued)**

File	Driverlib Function
sci.h	SCI_setFIFOInterruptLevel
sci.h	SCI_getFIFOInterruptLevel
sci.h	SCI_disableModule
sci.h	SCI_enableFIFO
sci.h	SCI_disableFIFO
sci.h	SCI_isFIFOEnabled
sci.h	SCI_resetTxFIFO
sci.h	SCI_resetChannels
sci.h	SCI_getTxFIFOStatus
sci.h	SCI_isTransmitterBusy
<b>FFRX</b>	
sci.c	SCI_enableInterrupt
sci.c	SCI_disableInterrupt
sci.c	SCI_getInterruptStatus
sci.c	SCI_clearInterruptStatus
sci.h	SCI_setFIFOInterruptLevel
sci.h	SCI_getFIFOInterruptLevel
sci.h	SCI_enableFIFO
sci.h	SCI_resetRxFIFO
sci.h	SCI_getRxFIFOStatus
sci.h	SCI_getOverflowStatus
sci.h	SCI_clearOverflowStatus
<b>FFCT</b>	
sci.h	SCI_lockAutobaud

## ***Inter-Integrated Circuit Module (I2C)***

---



---

This chapter describes the features and operation of the inter-integrated circuit (I2C) module. The I2C module provides an interface between one of these devices and devices compliant with the NXP Semiconductors Inter-IC bus (I2C bus) specification version 2.1, and connected by way of an I2C bus. External components attached to this 2-wire serial bus can transmit/receive 1 to 8-bit data to/from the device through the I2C module. This guide assumes the reader is familiar with the I2C bus specification.

---

**NOTE:** A unit of data transmitted or received by the I2C module can have fewer than 8 bits; however, for convenience, a unit of data is called a data byte throughout this document. The number of bits in a data byte is selectable via the BC bits of the mode register, I2CMR.

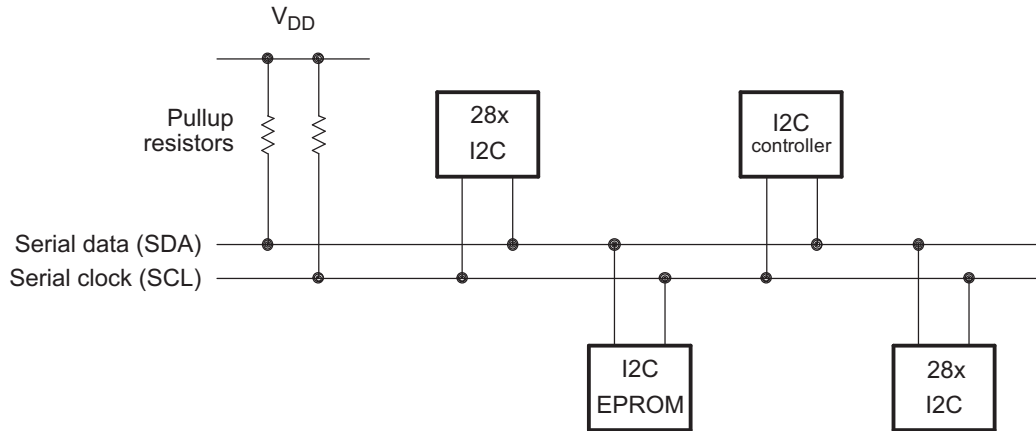
---

Topic	Page
<b>24.1 Introduction .....</b>	<b>2287</b>
<b>24.2 Configuring Device Pins .....</b>	<b>2290</b>
<b>24.3 I2C Module Operational Details.....</b>	<b>2291</b>
<b>24.4 Interrupt Requests Generated by the I2C Module.....</b>	<b>2298</b>
<b>24.5 Resetting or Disabling the I2C Module.....</b>	<b>2301</b>
<b>24.6 I2C Registers.....</b>	<b>2302</b>

## 24.1 Introduction

The I2C module supports any slave or master I2C-compatible device. [Figure 24-1](#) shows an example of multiple I2C modules connected for a two-way transfer from one device to other devices.

**Figure 24-1. Multiple I2C Modules Connected**



### 24.1.1 Features

The I2C module has the following features:

- Compliance with the NXP Semiconductors I2C bus specification (version 2.1):
  - Support for 8-bit format transfers
  - 7-bit and 10-bit addressing modes
  - General call
  - START byte mode
  - Support for multiple master-transmitters and slave-receivers
  - Support for multiple slave-transmitters and master-receivers
  - Combined master transmit/receive and receive/transmit mode
  - Data transfer rate from 10 kbps up to 400 kbps (Fast-mode)
- Receive FIFO and Transmitter FIFO (16-deep x 8-bit FIFO)
- Supports two ePIE interrupts:
  - I2Cx Interrupt – Any of the below events can be configured to generate an I2Cx interrupt:
    - Transmit-data ready
    - Receive-data ready
    - Register-access ready
    - No-acknowledgment received
    - Arbitration lost
    - Stop condition detected
    - Addressed as slave
  - I2Cx\_FIFO interrupts:
    - Transmit FIFO interrupt
    - Receive FIFO interrupt
- Module enable/disable capability
- Free data format mode

### 24.1.2 Features Not Supported

The I2C module does not support:

- High-speed mode (Hs-mode)
- CBUS-compatibility mode

### 24.1.3 Functional Overview

Each device connected to an I2C bus is recognized by a unique address. Each device can operate as either a transmitter or a receiver, depending on the function of the device. A device connected to the I2C bus can also be considered as the master or the slave when performing data transfers. A master device is the device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. During this transfer, any device addressed by this master is considered a slave. The I2C module supports the multi-master mode, in which one or more devices capable of controlling an I2C bus can be connected to the same I2C bus.

For data communication, the I2C module has a serial data pin (SDA) and a serial clock pin (SCL), as shown in [Section 24.6](#). These two pins carry information between the 28x device and other devices connected to the I2C bus. The SDA and SCL pins both are bidirectional. They each must be connected to a positive supply voltage using a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open-drain configuration to perform the required wired-AND function.

There are two major transfer techniques: .

- Standard Mode: Send exactly n data values, where n is a value you program in an I2C module register. See [Section 24.6](#) for more information.
- Repeat Mode: Keep sending data values until you use software to initiate a STOP condition or a new START condition. See *Registers* for RM bit information.

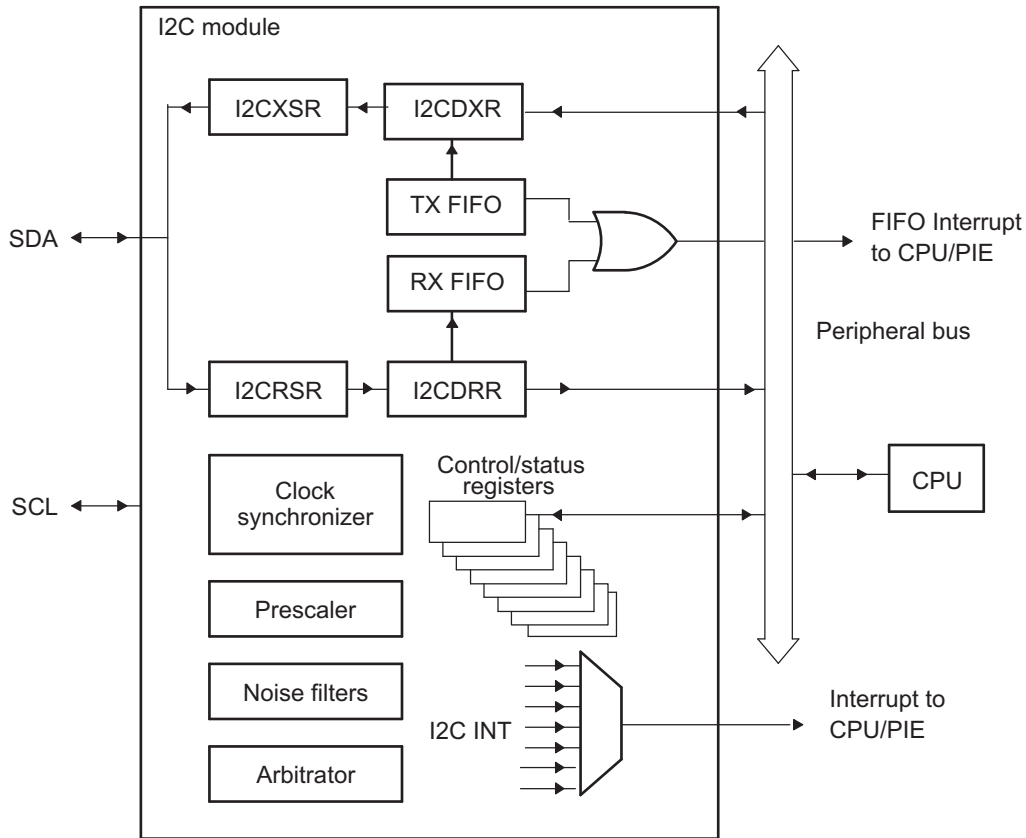
The I2C module consists of the following primary blocks:

- A serial interface: one data pin (SDA) and one clock pin (SCL)
- Data registers and FIFOs to temporarily hold receive data and transmit data traveling between the SDA pin and the CPU
- Control and status registers
- A peripheral bus interface to enable the CPU to access the I2C module registers and FIFOs.
- A clock synchronizer to synchronize the I2C input clock (from the device clock generator) and the clock on the SCL pin, and to synchronize data transfers with masters of different clock speeds
- A prescaler to divide down the input clock that is driven to the I2C module
- A noise filter on each of the two pins, SDA and SCL
- An arbitrator to handle arbitration between the I2C module (when it is a master) and another master
- Interrupt generation logic, so that an interrupt can be sent to the CPU
- FIFO interrupt generation logic, so that FIFO access can be synchronized to data reception and data transmission in the I2C module

[Figure 24-2](#) shows the four registers used for transmission and reception in non-FIFO mode. The CPU writes data for transmission to I2CDXR and reads received data from I2CDRR. When the I2C module is configured as a transmitter, data written to I2CDXR is copied to I2CXSR and shifted out on the SDA pin one bit at a time. When the I2C module is configured as a receiver, received data is shifted into I2CRSR and then copied to I2CDRR.



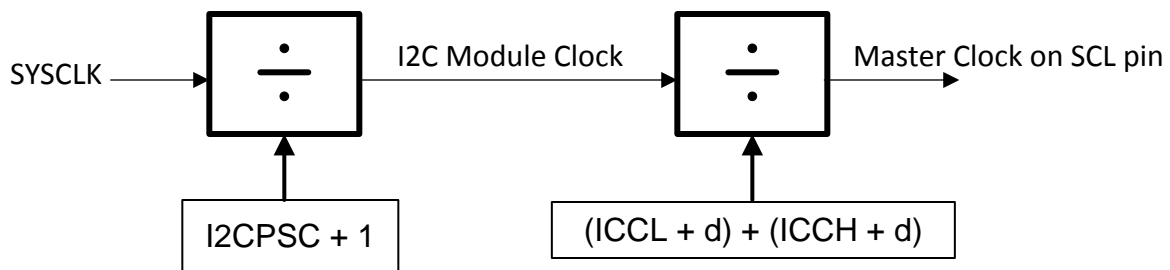
**Figure 24-2. I2C Module Conceptual Block Diagram**



**24.1.4 Clock Generation**

The I2C module clock determines the frequency at which the I2C module operates. A programmable prescaler in the I2C module divides down the SYSCLK to produce the I2C module clock and this I2C module clock is divided further to produce the I2C master clock on the SCL pin. Figure 24-3 shows the clock generation diagram for I2C module.

**Figure 24-3. Clocking Diagram for the I2C Module**



To specify the divide-down value, initialize the IPSC field of the prescaler register, I2CPSC. The resulting frequency is:

$$\text{I2C Module Clock (Fmod)} = \frac{\text{SYSCLK}}{(\text{I2CPSC} + 1)}$$

**NOTE:** To meet all of the I2C protocol timing specifications, the I2C module clock must be between 7 - 12 MHz.

The prescaler must be initialized only while the I2C module is in the reset state (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

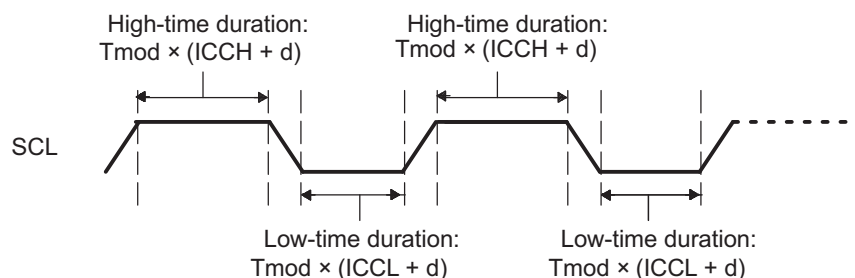
The master clock appears on the SCL pin when the I2C module is configured to be a master on the I2C bus. This clock controls the timing of communication between the I2C module and a slave. As shown in Figure 24-3, a second clock divider in the I2C module divides down the module clock to produce the master clock. The clock divider uses the ICCL value of I2CCLKL to divide down the low portion of the module clock signal and uses the ICCH value of I2CCLKH to divide down the high portion of the module clock signal. See Section 24.1.5 for the master clock frequency equation.

### 24.1.5 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)

As explained in Section 24.1.4, when the I2C module is a master, the I2C module clock is divided down further to use as the master clock on the SCL pin. As shown in Figure 24-4, the shape of the master clock depends on two divide-down values:

- ICCL in I2CCLKL. For each master clock cycle, ICCL determines the amount of time the signal is low.
- ICCH in I2CCLKH. For each master clock cycle, ICCH determines the amount of time the signal is high.

**Figure 24-4. The Roles of the Clock Divide-Down Values (ICCL and ICCH)**



#### 24.1.5.1 Formula for the Master Clock Period

The master clock period ( $T_{mst}$ ) is a multiple of the period of the I2C Module Clock ( $T_{mod}$ ):

$$\text{Master Clock period } (T_{mst}) = \frac{[(ICCH + d) + (ICCL + d)]}{\text{I2C Module Clock } (F_{mod})}$$

where  $d$  depends on the divide-down value IPSC, as shown in Table 24-1. IPSC is described in the I2CPSC register.

**Table 24-1. Dependency of Delay  $d$  on the Divide-Down Value IPSC**

IPSC	$d$
0	7
1	6
Greater than 1	5

## 24.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

### 24.3 I2C Module Operational Details

This section provides an overview of the I2C bus protocol and how it is implemented.

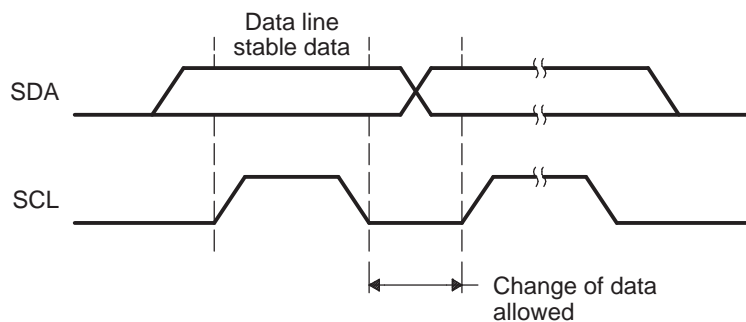
#### 24.3.1 Input and Output Voltage Levels

One clock pulse is generated by the master device for each data bit transferred. Due to a variety of different technology devices that can be connected to the I2C bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated level of  $V_{DD}$ . For details, see the data manual for your particular device.

#### 24.3.2 Data Validity

The data on SDA must be stable during the high period of the clock (see [Figure 24-5](#)). The high or low state of the data line, SDA, should change only when the clock signal on SCL is low.

**Figure 24-5. Bit Transfer on the I2C bus**



#### 24.3.3 Operating Modes

The I2C module has four basic operating modes to support data transfers as a master and as a slave. See [Table 24-2](#) for the names and descriptions of the modes.

If the I2C module is a master, it begins as a master-transmitter and typically transmits an address for a particular slave. When giving data to the slave, the I2C module must remain a master-transmitter. To receive data from a slave, the I2C module must be changed to the master-receiver mode.

If the I2C module is a slave, it begins as a slave-receiver and typically sends acknowledgment when it recognizes its slave address from a master. If the master will be sending data to the I2C module, the module must remain a slave-receiver. If the master has requested data from the I2C module, the module must be changed to the slave-transmitter mode.

**Table 24-2. Operating Modes of the I2C Module**

Operating Mode	Description
Slave-receiver modes	<p>The I2C module is a slave and receives data from a master.</p> <p>All slaves begin in this mode. In this mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the device is required (<math>RSFULL = 1</math> in <math>I2CSTR</math>) after a byte has been received. See <a href="#">Section 25.4</a> for more details.</p>

**Table 24-2. Operating Modes of the I2C Module (continued)**

Operating Mode	Description
Slave-transmitter mode	<p>The I2C module is a slave and transmits data to a master.</p> <p>This mode can be entered only from the slave-receiver mode; the I2C module must first receive a command from the master. When you are using any of the 7-bit/10-bit addressing formats, the I2C module enters its slave-transmitter mode if the slave address byte is the same as its own address (in I2COAR) and the master has transmitted <math>R/\overline{W} = 1</math>. As a slave-transmitter, the I2C module then shifts the serial data out on SDA with the clock pulses that are generated by the master. While a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the device is required (<math>XSMT = 0</math> in I2CSTR) after a byte has been transmitted. See section <a href="#">Section 25.4</a> for more details.</p>
Master-receiver mode	<p>The I2C module is a master and receives data from a slave.</p> <p>This mode can be entered only from the master-transmitter mode; the I2C module must first transmit a command to the slave. When you are using any of the 7-bit/10-bit addressing formats, the I2C module enters its master-receiver mode after transmitting the slave address byte and <math>R/\overline{W} = 1</math>. Serial data bits on SDA are shifted into the I2C module with the clock pulses generated by the I2C module on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the device is required (<math>RSFULL = 1</math> in I2CSTR) after a byte has been received.</p>
Master-transmitter modes	<p>The I2C module is a master and transmits control information and data to a slave.</p> <p>All masters begin in this mode. In this mode, data assembled in any of the 7-bit/10-bit addressing formats is shifted out on SDA. The bit shifting is synchronized with the clock pulses generated by the I2C module on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the device is required (<math>XSMT = 0</math> in I2CSTR) after a byte has been transmitted.</p>

To summarize, SCL will be held low in the following conditions:

- When an overrun condition is detected ( $RSFULL = 1$ ), in Slave-receiver mode.
- When an underflow condition is detected ( $XSMT = 0$ ), in Slave-transmitter mode.

I2C slave nodes have to accept and provide data when the I2C master node requests it.

- To release SCL in slave-receiver mode, read data from I2CDRR.
- To release SCL in slave-transmitter mode, write data to I2CDXR.
- To force a release without handling the data, reset the module using the I2CMDR.IRS bit.

**Table 24-3. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR**

RM	STT	STP	Bus Activity <sup>(1)</sup>	Description
0	0	0	None	No activity
0	0	1	P	STOP condition
0	1	0	S-A-D..(n)..D.	START condition, slave address, n data bytes (n = value in I2CCNT)
0	1	1	S-A-D..(n)..D-P	START condition, slave address, n data bytes, STOP condition (n = value in I2CCNT)
1	0	0	None	No activity
1	0	1	P	STOP condition
1	1	0	S-A-D-D-D.	Repeat mode transfer: START condition, slave address, continuous data transfers until STOP condition or next START condition
1	1	1	None	Reserved bit combination (No activity)

<sup>(1)</sup> S = START condition; A = Address; D = Data byte; P = STOP condition;

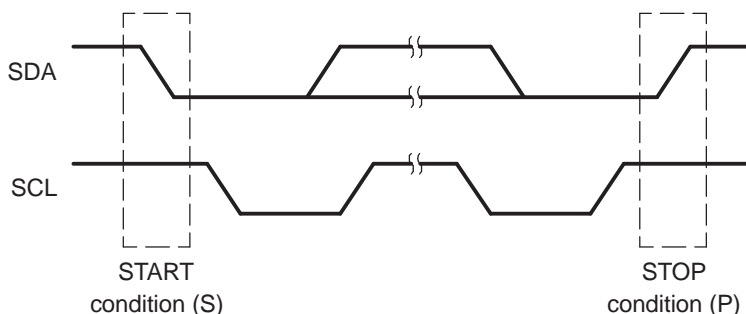
### 24.3.4 I2C Module START and STOP Conditions

START and STOP conditions can be generated by the I2C module when the module is configured to be a master on the I2C bus. As shown in [Figure 24-6](#):

- The START condition is defined as a high-to-low transition on the SDA line while SCL is high. A master drives this condition to indicate the start of a data transfer.

- The STOP condition is defined as a low-to-high transition on the SDA line while SCL is high. A master drives this condition to indicate the end of a data transfer.

**Figure 24-6. I2C Module START and STOP Conditions**



After a START condition and before a subsequent STOP condition, the I2C bus is considered busy, and the bus busy (BB) bit of I2CSTR is 1. Between a STOP condition and the next START condition, the bus is considered free, and BB is 0.

For the I2C module to start a data transfer with a START condition, the master mode bit (MST) and the START condition bit (STT) in I2CMDR must both be 1. For the I2C module to end a data transfer with a STOP condition, the STOP condition bit (STP) must be set to 1. When the BB bit is set to 1 and the STT bit is set to 1, a repeated START condition is generated. For a description of I2CMDR and its bits (including MST, STT, and STP), see *Registers Section 24.6*.

The I2C peripheral cannot detect a START or STOP condition while it is in reset (IRS = 0). The BB bit will remain in the cleared state (BB = 0) while the I2C peripheral is in reset (IRS = 0). When the I2C peripheral is taken out of reset (IRS set to 1) the BB bit will not correctly reflect the I2C bus status until a START or STOP condition is detected.

Follow these steps before initiating the first data transfer with I2C:

1. After taking the I2C peripheral out of reset by setting the IRS bit to 1, wait a period larger than the total time taken for the longest data transfer in the application. By waiting for a period of time after I2C comes out of reset, users can ensure that at least one START or STOP condition will have occurred on the I2C bus and been captured by the BB bit. After this period, the BB bit will correctly reflect the state of the I2C bus.
2. Check the BB bit and verify that BB = 0 (bus not busy) before proceeding.
3. Begin data transfers.

Not resetting the I2C peripheral in between transfers ensures that the BB bit reflects the actual bus status. If users must reset the I2C peripheral in between transfers, repeat steps 1 through 3 every time the I2C peripheral is taken out of reset.

### 24.3.5 Serial Data Formats

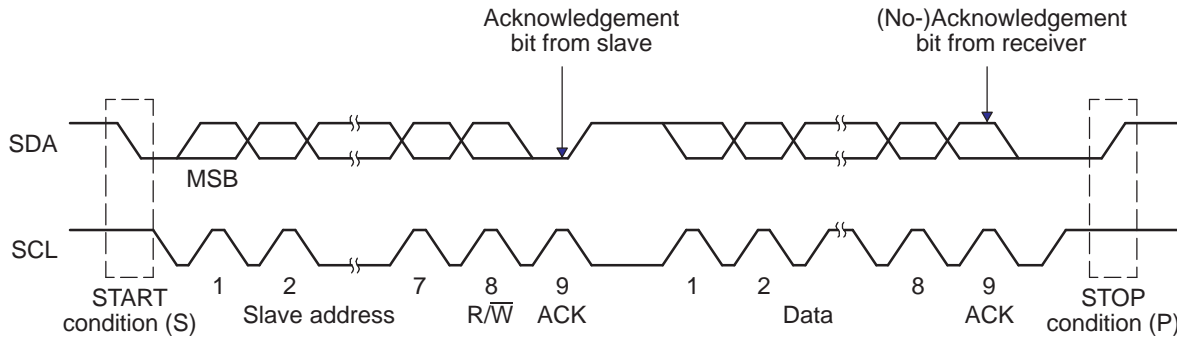
Figure 24-7 shows an example of a data transfer on the I2C bus. The I2C module supports 1 to 8-bit data values. In Figure 24-7, 8-bit data is transferred. Each bit put on the SDA line equates to 1 pulse on the SCL line, and the values are always transferred with the most significant bit (MSB) first. The number of data values that can be transmitted or received is unrestricted. The serial data format used in Figure 24-7 is the 7-bit addressing format. The I2C module supports the formats shown in Figure 24-8 through Figure 24-10 and described in the paragraphs that follow the figures.

---

**NOTE:** In Figure 24-7 through Figure 24-10, n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

---

**Figure 24-7. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown)**



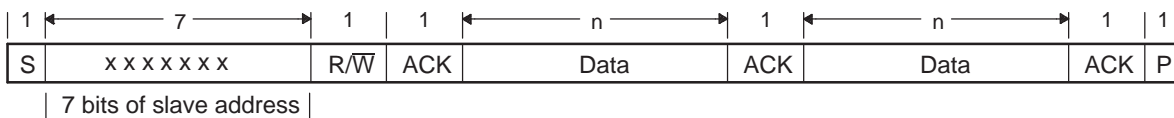
**24.3.5.1 7-Bit Addressing Format**

The 7-bit addressing format is the default format after reset. Disabling expanded address (I2CMDR.XA = 0) and free data format (I2CMDR.FDF = 0) enables 7-bit addressing format.

In this format (see Figure 24-8), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/W bit. R/W determines the direction of the data:

- R/W = 0: The I2C master writes (transmits) data to the addressed slave. This can be achieved by setting I2CMDR.TRX = 1 (Transmitter mode)
- R/W = 1: The I2C master reads (receives) data from the slave. This can be achieved by setting I2CMDR.TRX = 0 (Receiver mode)

**Figure 24-8. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)**



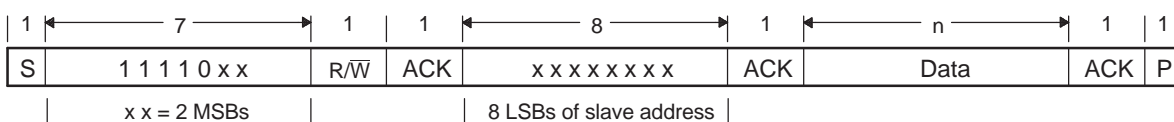
An extra clock cycle dedicated for acknowledgment (ACK) is inserted after each byte. If the ACK bit is inserted by the slave after the first byte from the master, it is followed by n bits of data from the transmitter (master or slave, depending on the R/W bit). n is a number from 1 to 8 determined by the bit count (BC) field of I2CMDR. After the data bits have been transferred, the receiver inserts an ACK bit.

**24.3.5.2 10-Bit Addressing Format**

The 10-bit addressing format can be enabled by setting expanded address (I2CMDR.XA = 1) and disabling free data format (I2CMDR.FDF = 0).

The 10-bit addressing format (see Figure 24-9) is similar to the 7-bit addressing format, but the master sends the slave address in two separate byte transfers. The first byte consists of 11110b, the two MSBs of the 10-bit slave address, and R/W. The second byte is the remaining 8 bits of the 10-bit slave address. The slave must send acknowledgment after each of the two byte transfers. Once the master has written the second byte to the slave, the master can either write data or use a repeated START condition to change the data direction. For more details about using 10-bit addressing, see the NXP Semiconductors I2C bus specification.

**Figure 24-9. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR)**

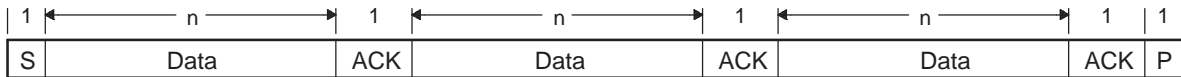


**24.3.5.3 Free Data Format**

The free data format can be enabled by setting I2CMDR.FDF = 1.

In this format (see [Figure 24-10](#)), the first byte after a START condition (S) is a data byte. An ACK bit is inserted after each data byte, which can be from 1 to 8 bits, depending on the BC field of I2CMDR. No address or data-direction bit is sent. Therefore, the transmitter and the receiver must both support the free data format, and the direction of the data must be constant throughout the transfer.

**Figure 24-10. I2C Module Free Data Format (FDF = 1 in I2CMDR)**



**NOTE:** The free data format is not supported in the digital loopback mode (I2CMDR.DLB = 1).

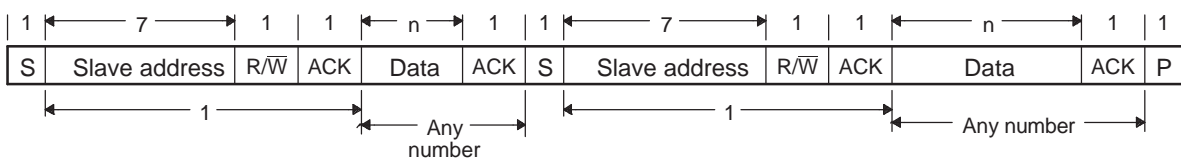
**Table 24-4. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR**

MST	FDF	I2C Module State	Function of TRX
0	0	In slave mode but not free data format mode	TRX is a don't care. Depending on the command from the master, the I2C module responds as a receiver or a transmitter.
0	1	In slave mode and free data format mode	The free data format mode requires that the I2C module remains the transmitter or the receiver throughout the transfer. TRX identifies the role of the I2C module:  TRX = 1: The I2C module is a transmitter. TRX = 0: The I2C module is a receiver.
1	0	In master mode but not free data format mode	TRX = 1: The I2C module is a transmitter. TRX = 0: The I2C module is a receiver.
1	1	In master mode and free data format mode	TRX = 0: The I2C module is a receiver. TRX = 1: The I2C module is a transmitter.

#### 24.3.5.4 Using a Repeated START Condition

I2C master can communicate with multiple slave addresses without having to give up control of the I2C bus by driving a STOP condition. This can be achieved by driving another START condition at the end of each data type. The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, and free data formats. [Figure 24-11](#) shows a repeated START condition in the 7-bit addressing format.

**Figure 24-11. Repeated START Condition (in This Case, 7-Bit Addressing Format)**



**NOTE:** In [Figure 24-11](#), n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

#### 24.3.6 NACK Bit Generation

When the I2C module is a receiver (master or slave), it can acknowledge or ignore bits sent by the transmitter. To ignore any new bits, the I2C module must send a no-acknowledge (NACK) bit during the acknowledge cycle on the bus. [Table 24-5](#) summarizes the various ways you can tell the I2C module to send a NACK bit.



**Table 24-5. Ways to Generate a NACK Bit**

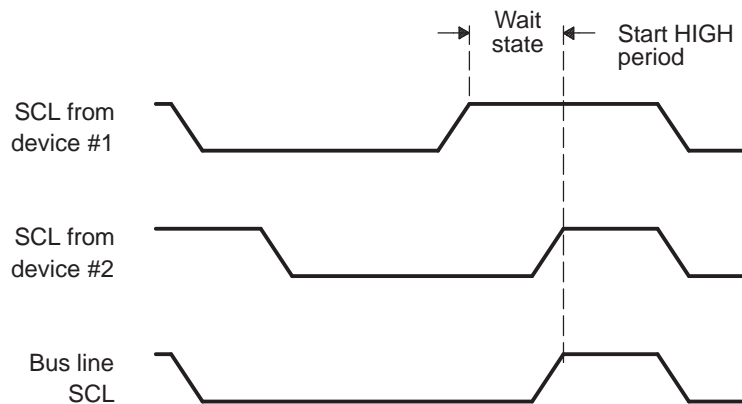
I2C Module Condition	NACK Bit Generation Options
Slave-receiver modes	<ul style="list-style-type: none"> <li>Allow an overrun condition (RSFULL = 1 in I2CSTR)</li> <li>Reset the module (IRS = 0 in I2CMDR)</li> <li>Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive</li> </ul>
Master-receiver mode AND Repeat mode (RM = 1 in I2CMDR)	<ul style="list-style-type: none"> <li>Generate a STOP condition (STP = 1 in I2CMDR)</li> <li>Reset the module (IRS = 0 in I2CMDR)</li> <li>Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive</li> </ul>
Master-receiver mode AND Nonrepeat mode (RM = 0 in I2CMDR)	<ul style="list-style-type: none"> <li>If STP = 1 in I2CMDR, allow the internal data counter to count down to 0 and thus force a STOP condition</li> <li>If STP = 0, make STP = 1 to generate a STOP condition</li> <li>Reset the module (IRS = 0 in I2CMDR). = 1 to generate a STOP condition</li> <li>Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive</li> </ul>

### 24.3.7 Clock Synchronization

Under normal conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, there are two or more masters and the clock must be synchronized so that the data output can be compared. Figure 24-12 illustrates the clock synchronization. The wired-AND property of SCL means that a device that first generates a low period on SCL overrules the other devices. At this high-to-low transition, the clock generators of the other devices are forced to start their own low period. The SCL is held low by the device with the longest low period. The other devices that finish their low periods must wait for SCL to be released, before starting their high periods. A synchronized signal on SCL is obtained, where the slowest device determines the length of the low period and the fastest device determines the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the wait state. In this way, a slave slows down a fast master and the slow device creates enough time to store a received byte or to prepare a byte to be transmitted.

**Figure 24-12. Synchronization of Two I2C Clock Generators During Arbitration**



### 24.3.8 Arbitration

If two or more master-transmitters attempt to start a transmission on the same bus at approximately the same time, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (SDA) by the competing transmitters. Figure 24-13 illustrates the arbitration procedure between two devices. The first master-transmitter that releases the SDA line high is overruled by another master-transmitter that drives the SDA low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

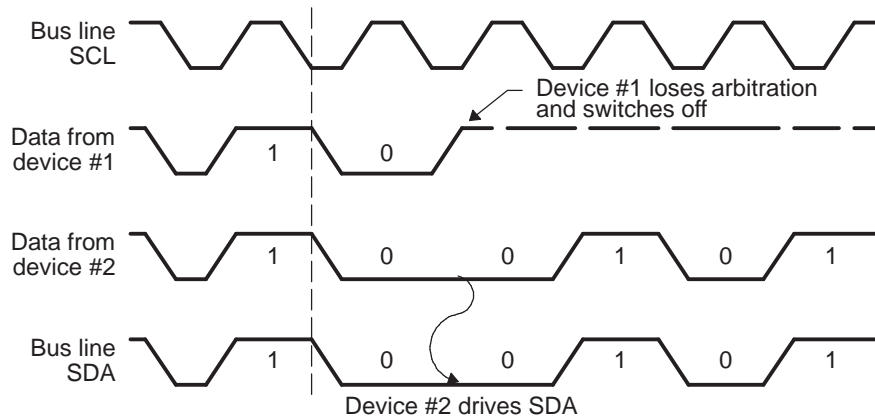


If the I2C module is the losing master, it switches to the slave-receiver mode, sets the arbitration lost (ARBL) flag, and generates the arbitration-lost interrupt request.

If during a serial transfer the arbitration procedure is still in progress when a repeated START condition or a STOP condition is transmitted to SDA, the master-transmitters involved must send the repeated START condition or the STOP condition at the same position in the format frame. Arbitration is not allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

**Figure 24-13. Arbitration Procedure Between Two Master-Transmitters**

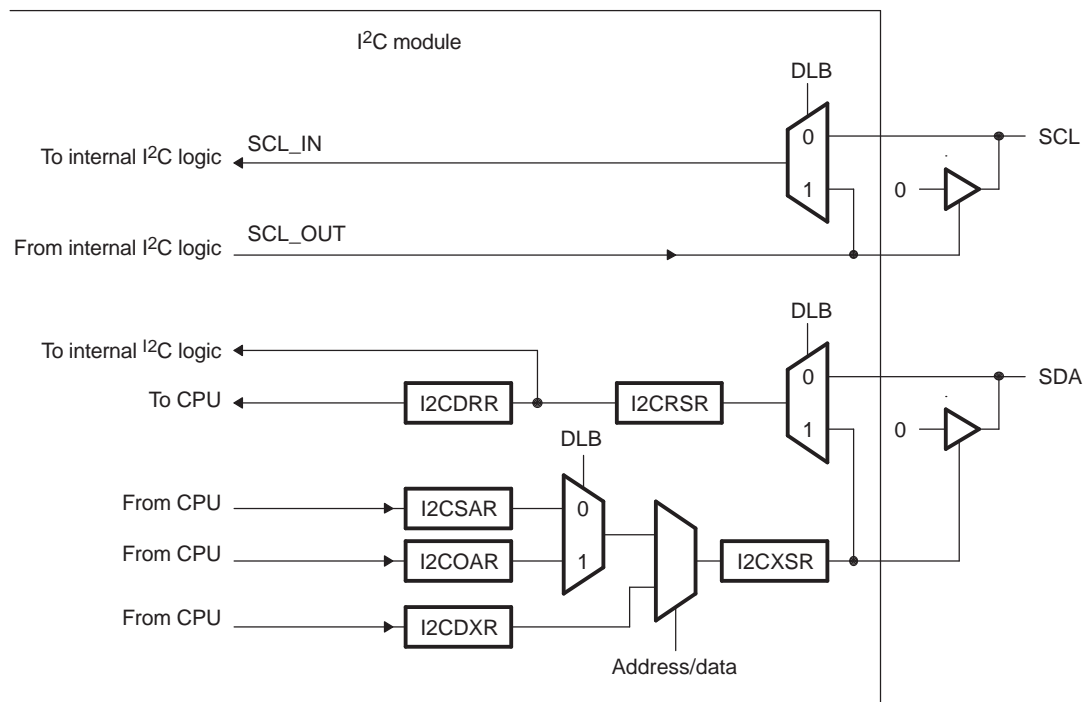


### 24.3.9 Digital Loopback Mode

The I2C module support a self-test mode called digital loopback, which is enabled by setting the DLB bit in the I2CMR register. In this mode, data transmitted out of the I2CDXR register is received in the I2CDRR register. The data follows an internal path, and takes  $n$  cycles to reach I2CDRR, where:

$$n = 8 * (\text{SYSCLK}) / (\text{I2C module clock (Fmod)})$$

The transmit clock and the receive clock are the same. The address seen on the external SDA pin is the address in the I2COAR register. [Figure 24-14](#) shows the signal routing in digital loopback mode.

**Figure 24-14. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit**


**NOTE:** The free data format (I2CMDR.FDF = 1) is not supported in digital loopback mode.

## 24.4 Interrupt Requests Generated by the I2C Module

Each I2C module can generate two CPU interrupts.

1. Basic I2C interrupt: Possible basic I2C interrupt sources which can trigger this interrupt are described in [Section 24.4.1](#).
2. I2C FIFO interrupt: Possible I2C FIFO interrupt sources which can trigger this interrupt are described in [Section 24.4.2](#)

### 24.4.1 Basic I2C Interrupt Requests

The I2C module generates the interrupt requests described in [Table 24-6](#). As shown in [Figure 24-15](#), all requests are multiplexed through an arbiter to a single I2C interrupt request to the CPU. Each interrupt request has a flag bit in the status register (I2CSTR) and an enable bit in the interrupt enable register (I2CIER). When one of the specified events occurs, its flag bit is set. If the corresponding enable bit is 0, the interrupt request is blocked. If the enable bit is 1, the request is forwarded to the CPU as an I2C interrupt.

The I2C interrupt is one of the maskable interrupts of the CPU. As with any maskable interrupt request, if it is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (I2CINT1A\_ISR). The I2CINT1A\_ISR for the I2C interrupt can determine the interrupt source by reading the interrupt source register, I2CISRC. Then the I2CINT1A\_ISR can branch to the appropriate subroutine.

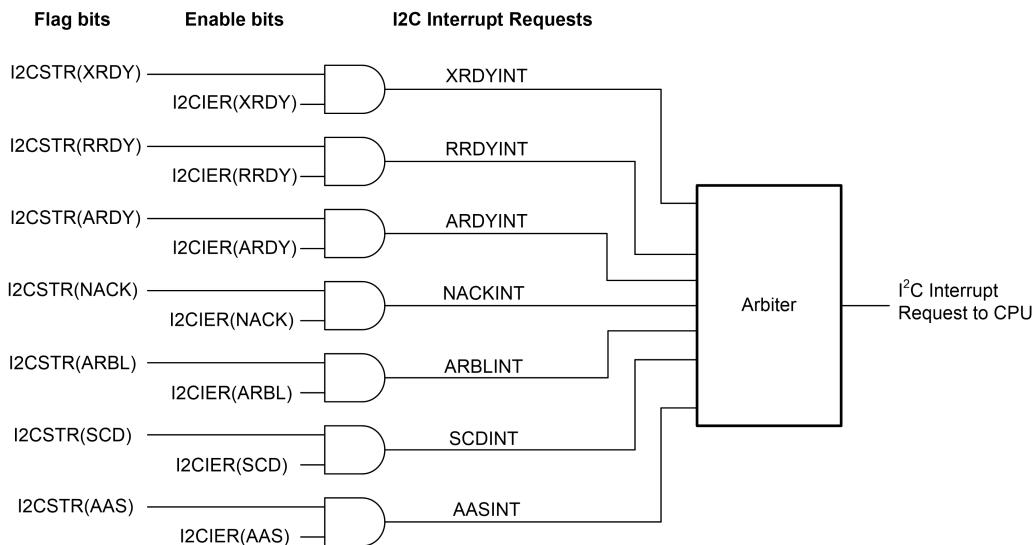
After the CPU reads I2CISRC, the following events occur:

1. The flag for the source interrupt is cleared in I2CSTR. Exception: The ARDY, RRDY, and XRDY bits in I2CSTR are not cleared when I2CISRC is read. To clear one of these bits, write a 1 to it.
2. The arbiter determines which of the remaining interrupt requests has the highest priority, writes the code for that interrupt to I2CISRC, and forwards the interrupt request to the CPU.

**Table 24-6. Descriptions of the Basic I2C Interrupt Requests**

I2C Interrupt Request	Interrupt Source
XRDYINT	<p>Transmit ready condition: The data transmit register (I2CDXR) is ready to accept new data because the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR).</p> <p>As an alternative to using XRDYINT, the CPU can poll the XRDY bit of the status register, I2CSTR. XRDYINT should not be used when in FIFO mode. Use the FIFO interrupts instead.</p>
RRDYINT	<p>Receive ready condition: The data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR.</p> <p>As an alternative to using RRDYINT, the CPU can poll the RRDY bit of I2CSTR. RRDYINT should not be used when in FIFO mode. Use the FIFO interrupts instead.</p>
ARDYINT	<p>Register-access ready condition: The I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used.</p> <p>The specific events that generate ARDYINT are the same events that set the ARDY bit of I2CSTR.</p> <p>As an alternative to using ARDYINT, the CPU can poll the ARDY bit.</p>
NACKINT	<p>No-acknowledgment condition: The I2C module is configured as a master-transmitter and did not receive acknowledgment from the slave-receiver.</p> <p>As an alternative to using NACKINT, the CPU can poll the NACK bit of I2CSTR.</p>
ARBLINT	<p>Arbitration-lost condition: The I2C module has lost an arbitration contest with another master-transmitter.</p> <p>As an alternative to using ARBLINT, the CPU can poll the ARBL bit of I2CSTR.</p>
SCDINT	<p>Stop condition detected: A STOP condition was detected on the I2C bus.</p> <p>As an alternative to using SCDINT, the CPU can poll the SCD bit of the status register, I2CSTR.</p>
AASINT	<p>Addressed as slave condition: The I2C has been addressed as a slave device by another master on the I2C bus.</p> <p>As an alternative to using AASINT, the CPU can poll the AAS bit of the status register, I2CSTR.</p>

**Figure 24-15. Enable Paths of the I2C Interrupt Requests**



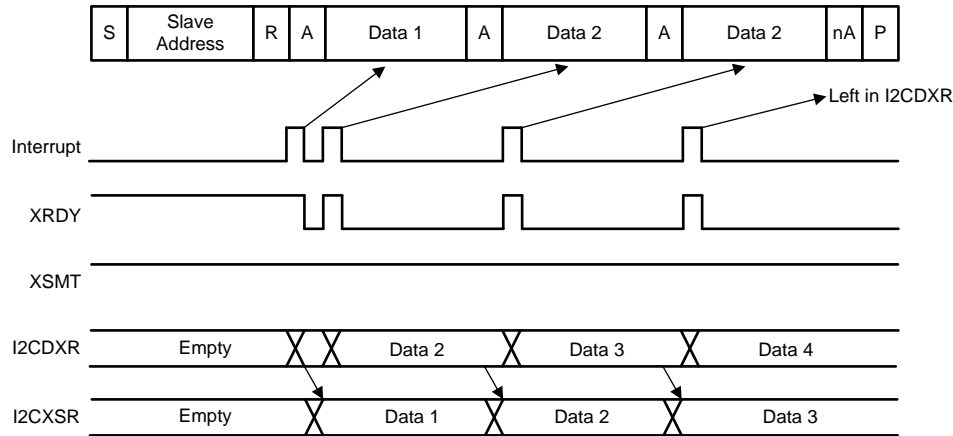
The normal transmit interrupt timing makes it possible for stale data to remain in the transmit buffer if a transaction is aborted in the middle of a byte. To avoid this, set the FCM bit in the I2CEMDR register. When this bit is set, the transmit data ready interrupt is generated only when data is required for a bus transaction. In master mode, the interrupt is first generated when the ACK of the address byte is received. In slave mode, the interrupt is first generated when the address is matched. Further interrupts are generated when the data is ACKed. In this mode XRDY is asserted at the same time as the transmit ready interrupt.

The I2C module has a backwards compatibility bit (BC) in the I2CEMDR register. The timing diagram in [Figure 24-16](#) demonstrates the effect the backwards compatibility bit has on I2C module registers and interrupts when configured as a slave-transmitter.

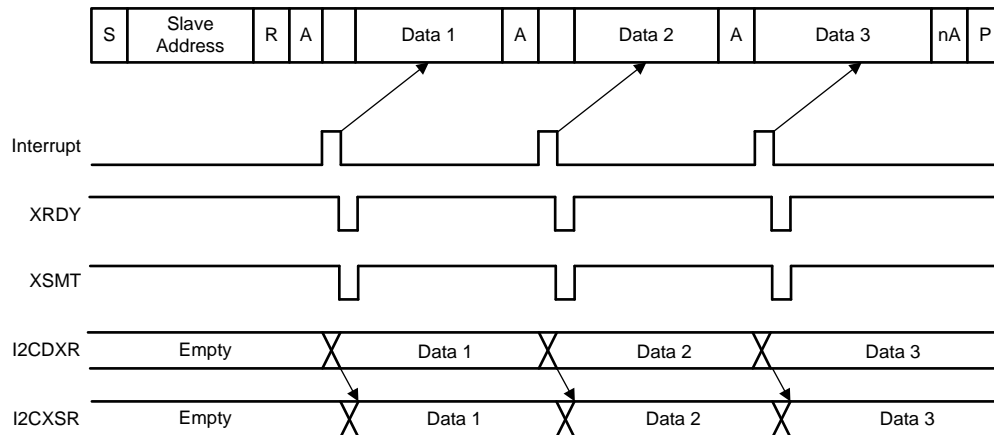
**Figure 24-16. Backwards Compatibility Mode Bit, Slave Transmitter**

Slave-Transmitter

b) BC = 1



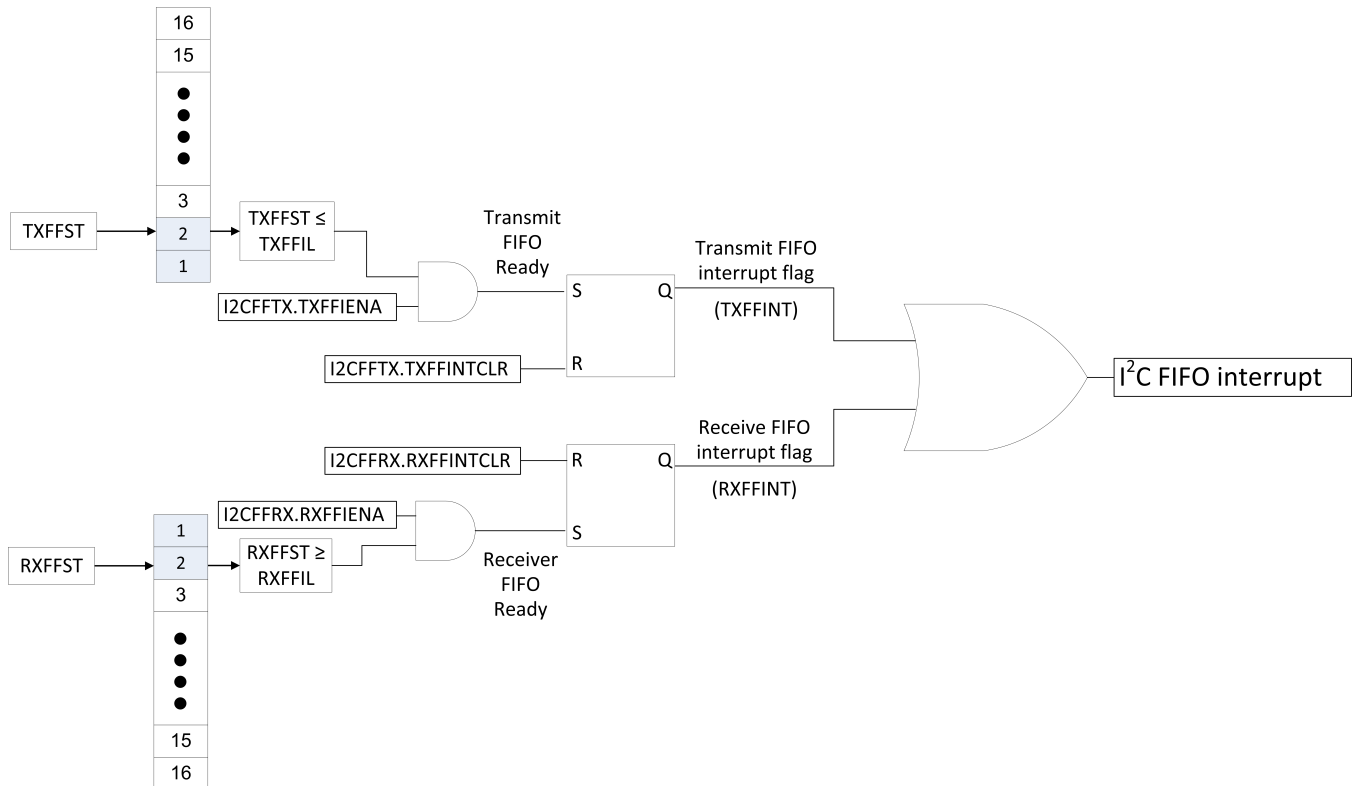
b) BC = 0



### 24.4.2 I2C FIFO Interrupts

In addition to the seven basic I2C interrupts, the transmit and receive FIFOs each contain the ability to generate an interrupt (I2CINT2A). The transmit FIFO can be configured to generate an interrupt after transmitting a defined number of bytes, up to 16. The receive FIFO can be configured to generate an interrupt after receiving a defined number of bytes, up to 16. These two interrupt sources are ORed together into a single maskable CPU interrupt. Figure 24-17 shows the structure of I2C FIFO interrupt. The interrupt service routine can then read the FIFO interrupt status flags to determine from which source the interrupt came. See the I2C transmit FIFO register (I2CFFTX) and the I2C receive FIFO register (I2CFFRX) descriptions.

Figure 24-17. I2C\_FIFO\_interrupt



## 24.5 Resetting or Disabling the I2C Module

You can reset or disable the I2C module in two ways:

- Write 0 to the I2C reset bit (IRS) in the I2C mode register (I2CMDR). All status bits (in I2CSTR) are forced to their default values, and the I2C module remains disabled until IRS is changed to 1. The SDA and SCL pins are in the high-impedance state.
- Initiate a device reset by driving the  $\overline{XRS}$  pin low. The entire device is reset and is held in the reset state until you drive the pin high. When the  $\overline{XRS}$  pin is released, all I2C module registers are reset to their default values. The IRS bit is forced to 0, which resets the I2C module. The I2C module stays in the reset state until you write 1 to IRS.

The IRS must be 0 while you configure or reconfigure the I2C module. Forcing IRS to 0 can be used to save power and to clear error conditions.

## 24.6 I2C Registers

This section describes the C28x I2C Module Registers.

### 24.6.1 I2C Base Addresses

**Table 24-7. I2C Base Address Table**

Device Registers	Register Name	Start Address	End Address
I2caRegs	I2C_REGS	0x0000_7300	0x0000_733F

### 24.6.2 I2C\_REGS Registers

Table 24-8 lists the I2C\_REGS registers. All register offset addresses not listed in Table 24-8 should be considered as reserved locations and the register contents should not be modified.

**Table 24-8. I2C\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	I2COAR	I2C Own address		<a href="#">Go</a>
1h	I2CIER	I2C Interrupt Enable		<a href="#">Go</a>
2h	I2CSTR	I2C Status		<a href="#">Go</a>
3h	I2CCLKL	I2C Clock low-time divider		<a href="#">Go</a>
4h	I2CCLKH	I2C Clock high-time divider		<a href="#">Go</a>
5h	I2CCNT	I2C Data count		<a href="#">Go</a>
6h	I2CDRR	I2C Data receive		<a href="#">Go</a>
7h	I2CSAR	I2C Slave address		<a href="#">Go</a>
8h	I2CDXR	I2C Data Transmit		<a href="#">Go</a>
9h	I2CMODR	I2C Mode		<a href="#">Go</a>
Ah	I2CISRC	I2C Interrupt Source		<a href="#">Go</a>
Bh	I2CEMDR	I2C Extended Mode		<a href="#">Go</a>
Ch	I2CPSC	I2C Prescaler		<a href="#">Go</a>
20h	I2CFFTX	I2C FIFO Transmit		<a href="#">Go</a>
21h	I2CFFRX	I2C FIFO Receive		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 24-9 shows the codes that are used for access types in this section.

**Table 24-9. I2C\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1C	W1C	Write 1 to clear
W1S	W1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**24.6.2.1 I2COAR Register (Offset = 0h) [reset = 0h]**

I2COAR is shown in [Figure 24-18](#) and described in [Table 24-10](#).

Return to the [Summary Table](#).

The I2C own address register (I2COAR) is a 16-bit register. The I2C module uses this register to specify its own slave address, which distinguishes it from other slaves connected to the I2C-bus. If the 7-bit addressing mode is selected (XA = 0 in I2CMDR), only bits 6-0 are used write 0s to bits 9-7.

**Figure 24-18. I2COAR Register**

15	14	13	12	11	10	9	8
RESERVED						OAR	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
OAR							
R/W-0h							

**Table 24-10. I2COAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OAR	R/W	0h	<p>In 7-bit addressing mode (XA = 0 in I2CMDR): 00h-7Fh Bits 6-0 provide the 7-bit slave address of the I2C module. Write 0s to bits 9-7.</p> <p>In 10-bit addressing mode (XA = 1 in I2CMDR): 000h-3FFh Bits 9-0 provide the 10-bit slave address of the I2C module. Reset type: SYSRSn</p>



### 24.6.2.2 I2CIER Register (Offset = 1h) [reset = 0h]

I2CIER is shown in [Figure 24-19](#) and described in [Table 24-11](#).

Return to the [Summary Table](#).

I2CIER is used by the CPU to individually enable or disable I2C interrupt requests.

**Figure 24-19. I2CIER Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	AAS	SCD	XRDY	RRDY	ARDY	NACK	ARBL
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 24-11. I2CIER Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	AAS	R/W	0h	Addressed as slave interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
5	SCD	R/W	0h	Stop condition detected interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
4	XRDY	R/W	0h	Transmit-data-ready interrupt enable bit. This bit should not be set when using FIFO mode. Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
3	RRDY	R/W	0h	Receive-data-ready interrupt enable bit. This bit should not be set when using FIFO mode. Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
2	ARDY	R/W	0h	Register-access-ready interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
1	NACK	R/W	0h	No-acknowledgment interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
0	ARBL	R/W	0h	Arbitration-lost interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled

### 24.6.2.3 I2CSTR Register (Offset = 2h) [reset = 0h]

I2CSTR is shown in [Figure 24-20](#) and described in [Table 24-12](#).

Return to the [Summary Table](#).

The I2C status register (I2CSTR) is a 16-bit register used to determine which interrupt has occurred and to read status information.

**Figure 24-20. I2CSTR Register**

15		14		13		12		11		10		9		8	
RESERVED	SDIR	NACKSNT	BB	RSFULL	XSMT	AAS	AD0								
R-0h	R/W1C-0h	R/W1C-0h	R-0h	R-0h	R-0h	R-0h	R-0h								
7		6		5		4		3		2		1		0	
RESERVED	BYTESENT	SCD	XRDY	RRDY	ARDY	NACK	ARBL								
R-0h	R/W1C-0h	R/W1C-0h	R-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h								

**Table 24-12. I2CSTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	SDIR	R/W1C	0h	Slave direction bit Reset type: SYSRSn 0h (R/W) = I2C is not addressed as a slave transmitter. SDIR is cleared by one of the following events: - It is manually cleared. To clear this bit, write a 1 to it. - Digital loopback mode is enabled. - A START or STOP condition occurs on the I2C bus. 1h (R/W) = I2C is addressed as a slave transmitter.
13	NACKSNT	R/W1C	0h	NACK sent bit. This bit is used when the I2C module is in the receiver mode. One instance in which NACKSNT is affected is when the NACK mode is used (see the description for NACKMOD in Reset type: SYSRSn 0h (R/W) = NACK not sent. NACKSNT bit is cleared by any one of the following events: - It is manually cleared. To clear this bit, write a 1 to it. - The I2C module is reset (either when 0 is written to the IRS bit of I2CMDR or when the whole device is reset). 1h (R/W) = NACK sent: A no-acknowledge bit was sent during the acknowledge cycle on the I2C-bus.
12	BB	R	0h	Bus busy bit. BB indicates whether the I2C-bus is busy or is free for another data transfer. See the paragraph following the table for more information Reset type: SYSRSn 0h (R/W) = Bus free. BB is cleared by any one of the following events: - The I2C module receives or transmits a STOP bit (bus free). - The I2C module is reset. 1h (R/W) = Bus busy: The I2C module has received or transmitted a START bit on the bus.

**Table 24-12. I2CSTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	RSFULL	R	0h	<p>Receive shift register full bit.</p> <p>RSFULL indicates an overrun condition during reception. Overrun occurs when new data is received into the shift register (I2CRSR) and the old data has not been read from the receive register (I2CDRR). As new bits arrive from the SDA pin, they overwrite the bits in I2CRSR. The new data will not be copied to ICDRR until the previous data is read.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No overrun detected. RSFULL is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- I2CDRR is read is read by the CPU. Emulator reads of the I2CDRR do not affect this bit.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = Overrun detected</p>
10	XSMT	R	0h	<p>Transmit shift register empty bit.</p> <p>XSMT = 0 indicates that the transmitter has experienced underflow. Underflow occurs when the transmit shift register (I2CXSR) is empty but the data transmit register (I2CDXR) has not been loaded since the last I2CDXR-to-I2CXSR transfer. The next I2CDXR-to-I2CXSR transfer will not occur until new data is in I2CDXR. If new data is not transferred in time, the previous data may be re-transmitted on the SDA pin.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Underflow detected (empty)</p> <p>1h (R/W) = No underflow detected (not empty). XSMT is set by one of the following events:</p> <ul style="list-style-type: none"> <li>- Data is written to I2CDXR.</li> <li>- The I2C module is reset</li> </ul>
9	AAS	R	0h	<p>Addressed-as-slave bit</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = In the 7-bit addressing mode, the AAS bit is cleared when receiving a NACK, a STOP condition, or a repeated START condition. In the 10-bit addressing mode, the AAS bit is cleared when receiving a NACK, a STOP condition, or by a slave address different from the I2C peripheral's own slave address.</p> <p>1h (R/W) = The I2C module has recognized its own slave address or an address of all zeros (general call).</p>
8	ADO	R	0h	<p>Address 0 bits</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = ADO has been cleared by a START or STOP condition.</p> <p>1h (R/W) = An address of all zeros (general call) is detected.</p>
7	RESERVED	R	0h	Reserved
6	BYTESENT	R/W1C	0h	<p>Byte Transmit over indication.</p> <p>BYTESENT is set when the master/slave has successfully sent the byte on SCL/SDA lines. This is diagnostic register which needs to be explicitly cleared by Software. In case not cleared the stale status would keep reflecting as no automated clear incorporated to avoid corner conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The I2C module has not finished transmitting the next data byte. BYTESENT is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- It is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = The I2C module has completed the transmission of a byte.</p>

**Table 24-12. I2CSTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	SCD	R/W1C	0h	<p>Stop condition detected bit.</p> <p>SCD is set when the I2C sends or receives a STOP condition. The I2C module delays clearing of the I2CMDR[STP] bit until the SCD bit is set.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = STOP condition not detected since SCD was last cleared. SCD is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- I2CISRC is read by the CPU when it contains the value 110b (stop condition detected). Emulator reads of the I2CISRC do not affect this bit.</li> <li>- SCD is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = A STOP condition has been detected on the I2C bus.</p>
4	XRDY	R	0h	<p>Transmit-data-ready interrupt flag bit. When not in FIFO mode, XRDY indicates that the data transmit register (I2CDXR) is ready to accept new data.</p> <p>FCM=0 : When the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR). The CPU can poll XRDY or use the XRDY interrupt request. When in FIFO mode, use TXFFINT instead.</p> <p>FCM=1: XRDY is asserted only when next data is required. It gets deasserted with write to I2CDXR. Both Polling and interrupt based data transfers are allowed in the FCM mode.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = I2CDXR not ready. XRDY is cleared when data is written to I2CDXR.</p> <p>1h (R/W) = I2CDXR ready: Data has been copied from I2CDXR to I2CXSR.</p> <p>XRDY is also forced to 1 when the I2C module is reset.</p>
3	RRDY	R/W1C	0h	<p>Receive-data-ready interrupt flag bit.</p> <p>When not in FIFO mode, RRDY indicates that the data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR. The CPU can poll RRDY or use the RRDY interrupt request. When in FIFO mode, use RXFFINT instead.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = I2CDRR not ready. RRDY is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- I2CDRR is read by the CPU. Emulator reads of the I2CDRR do not affect this bit.</li> <li>- RRDY is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = I2CDRR ready: Data has been copied from I2CRSR to I2CDRR.</p>

**Table 24-12. I2CSTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	ARDY	R/W1C	0h	<p>Register-access-ready interrupt flag bit (only Applicable when the I2C module is in the master mode).</p> <p>ARDY indicates that the I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used. The CPU can poll ARDY or use the ARDY interrupt request</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The registers are not ready to be accessed. ARDY is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- The I2C module starts using the current register contents.</li> <li>- ARDY is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = The registers are ready to be accessed.</p> <p>In the nonrepeat mode (RM = 0 in I2CMDR): If STP = 0 in I2CMDR, the ARDY bit is set when the internal data counter counts down to 0. If STP = 1, ARDY is not affected (instead, the I2C module generates a STOP condition when the counter reaches 0).</p> <p>In the repeat mode (RM = 1): ARDY is set at the end of each byte transmitted from I2CDXR.</p>
1	NACK	R/W1C	0h	<p>No-acknowledgment interrupt flag bit.</p> <p>NACK applies when the I2C module is a transmitter (master or slave). NACK indicates whether the I2C module has detected an acknowledge bit (ACK) or a noacknowledge bit (NACK) from the receiver. The CPU can poll NACK or use the NACK interrupt request.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = ACK received/NACK not received. This bit is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- An acknowledge bit (ACK) has been sent by the receiver.</li> <li>- NACK is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The CPU reads the interrupt source register (I2CISRC) and the register contains the code for a NACK interrupt. Emulator reads of the I2CISRC do not affect this bit.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = NACK bit received. The hardware detects that a no-acknowledge (NACK) bit has been received.</p> <p>Note: While the I2C module performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgment.</p>
0	ARBL	R/W1C	0h	<p>Arbitration-lost interrupt flag bit (only applicable when the I2C module is a master-transmitter).</p> <p>ARBL primarily indicates when the I2C module has lost an arbitration contest with another mastertransmitter. The CPU can poll ARBL or use the ARBL interrupt request.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Arbitration not lost. AL is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- AL is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The CPU reads the interrupt source register (I2CISRC) and the register contains the code for an AL interrupt. Emulator reads of the I2CISRC do not affect this bit.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = Arbitration lost. AL is set by any one of the following events:</p> <ul style="list-style-type: none"> <li>- The I2C module senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously.</li> <li>- The I2C module attempts to start a transfer while the BB (bus busy) bit is set to 1.</li> </ul> <p>When AL becomes 1, the MST and STP bits of I2CMDR are cleared, and the I2C module becomes a slave-receiver.</p>

**24.6.2.4 I2CCLKL Register (Offset = 3h) [reset = 0h]**

I2CCLKL is shown in [Figure 24-21](#) and described in [Table 24-13](#).

Return to the [Summary Table](#).

I2C Clock low-time divider

**Figure 24-21. I2CCLKL Register**

15	14	13	12	11	10	9	8
I2CCLKL							
R/W-0h							
7	6	5	4	3	2	1	0
I2CCLKL							
R/W-0h							

**Table 24-13. I2CCLKL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	I2CCLKL	R/W	0h	<p>Clock low-time divide-down value.</p> <p>To produce the low time duration of the master clock, the period of the module clock is multiplied by (ICCL + d). d is an adjustment factor based on the prescaler. See the Clock Divider Registers section of the Introduction for details.</p> <p>Note: These bits must be set to a non-zero value for proper I2C clock generation.</p> <p>Reset type: SYSRSn</p>

### 24.6.2.5 I2CCLKH Register (Offset = 4h) [reset = 0h]

I2CCLKH is shown in [Figure 24-22](#) and described in [Table 24-14](#).

Return to the [Summary Table](#).

I2C Clock high-time divider

**Figure 24-22. I2CCLKH Register**

15	14	13	12	11	10	9	8
I2CCLKH							
R/W-0h							
7	6	5	4	3	2	1	0
I2CCLKH							
R/W-0h							

**Table 24-14. I2CCLKH Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	I2CCLKH	R/W	0h	<p>Clock high-time divide-down value.</p> <p>To produce the high time duration of the master clock, the period of the module clock is multiplied by (ICCL + d). d is an adjustment factor based on the prescaler. See the Clock Divider Registers section of the Introduction for details.</p> <p>Note: These bits must be set to a non-zero value for proper I2C clock generation.</p> <p>Reset type: SYSRSn</p>

### 24.6.2.6 I2CCNT Register (Offset = 5h) [reset = 0h]

I2CCNT is shown in [Figure 24-23](#) and described in [Table 24-15](#).

Return to the [Summary Table](#).

I2CCNT is a 16-bit register used to indicate how many data bytes to transfer when the I2C module is configured as a transmitter, or to receive when configured as a master receiver. In the repeat mode (RM = 1), I2CCNT is not used.

The value written to I2CCNT is copied to an internal data counter. The internal data counter is decremented by 1 for each byte transferred (I2CCNT remains unchanged). If a STOP condition is requested in the master mode (STP = 1 in I2CMDR), the I2C module terminates the transfer with a STOP condition when the countdown is complete (that is, when the last byte has been transferred).

**Figure 24-23. I2CCNT Register**

15	14	13	12	11	10	9	8
I2CCNT							
R/W-0h							
7	6	5	4	3	2	1	0
I2CCNT							
R/W-0h							

**Table 24-15. I2CCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	I2CCNT	R/W	0h	Data count value. I2CCNT indicates the number of data bytes to transfer or receive. The value in I2CCNT is a don't care when the RM bit in I2CMDR is set to 1. The start value loaded to the internal data counter is 65536. The start value loaded to internal data counter is 1-65535. Reset type: SYSRSn



### 24.6.2.7 I2CDRR Register (Offset = 6h) [reset = 0h]

I2CDRR is shown in [Figure 24-24](#) and described in [Table 24-16](#).

Return to the [Summary Table](#).

I2CDRR is a 16-bit register used by the CPU to read received data. The I2C module can receive a data byte with 1 to 8 bits. The number of bits is selected with the bit count (BC) bits in I2CMDR. One bit at a time is shifted in from the SDA pin to the receive shift register (I2CRSR). When a complete data byte has been received, the I2C module copies the data byte from I2CRSR to I2CDRR. The CPU cannot access I2CRSR directly.

If a data byte with fewer than 8 bits is in I2CDRR, the data value is right-justified, and the other bits of I2CDRR(7-0) are undefined. For example, if BC = 011 (3-bit data size), the receive data is in I2CDRR(2-0), and the content of I2CDRR(7-3) is undefined.

When in the receive FIFO mode, the I2CDRR register acts as the receive FIFO buffer.

**Figure 24-24. I2CDRR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
DATA							
R-0h							

**Table 24-16. I2CDRR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	DATA	R	0h	Receive data Reset type: SYSRSn

### 24.6.2.8 I2CSAR Register (Offset = 7h) [reset = 3FFh]

I2CSAR is shown in [Figure 24-25](#) and described in [Table 24-17](#).

Return to the [Summary Table](#).

The I2C slave address register (I2CSAR) is a 16-bit register for storing the next slave address that will be transmitted by the I2C module when it is a master. The SAR field of I2CSAR contains a 7-bit or 10-bit slave address. When the I2C module is not using the free data format (FDF = 0 in I2CMDR), it uses this address to initiate data transfers with a slave, or slaves. When the address is nonzero, the address is for a particular slave. When the address is 0, the address is a general call to all slaves. If the 7-bit addressing mode is selected (XA = 0 in I2CMDR), only bits 6-0 of I2CSAR are used write 0s to bits 9-7.

**Figure 24-25. I2CSAR Register**

15	14	13	12	11	10	9	8
RESERVED						SAR	
R-0h						R/W-3FFh	
7	6	5	4	3	2	1	0
SAR							
R/W-3FFh							

**Table 24-17. I2CSAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	SAR	R/W	3FFh	<p>In 7-bit addressing mode (XA = 0 in I2CMDR): 00h-7Fh Bits 6-0 provide the 7-bit slave address that the I2C module transmits when it is in the master-transmitter mode. Write 0s to bits 9-7.</p> <p>In 10-bit addressing mode (XA = 1 in I2CMDR): 000h-3FFh Bits 9-0 provide the 10-bit slave address that the I2C module transmits when it is in the master transmitter mode. Reset type: SYSRSn</p>

### 24.6.2.9 I2CDXR Register (Offset = 8h) [reset = 0h]

I2CDXR is shown in [Figure 24-26](#) and described in [Table 24-18](#).

Return to the [Summary Table](#).

The CPU writes transmit data to I2CDXR. This 16-bit register accepts a data byte with 1 to 8 bits. Before writing to I2CDXR, specify how many bits are in a data byte by loading the appropriate value into the bit count (BC) bits of I2CMDR. When writing a data byte with fewer than 8 bits, make sure the value is right-aligned in I2CDXR.

After a data byte is written to I2CDXR, the I2C module copies the data byte to the transmit shift register (I2CXSR). The CPU cannot access I2CXSR directly. From I2CXSR, the I2C module shifts the data byte out on the SDA pin, one bit at a time.

When in the transmit FIFO mode, the I2CDXR register acts as the transmit FIFO buffer.

**Figure 24-26. I2CDXR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
DATA							
R/W-0h							

**Table 24-18. I2CDXR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	DATA	R/W	0h	Transmit data Reset type: SYSRSn

**24.6.2.10 I2CMR Register (Offset = 9h) [reset = 0h]**

I2CMR is shown in [Figure 24-27](#) and described in [Table 24-19](#).

Return to the [Summary Table](#).

The I2C mode register (I2CMR) is a 16-bit register that contains the control bits of the I2C module.

**Figure 24-27. I2CMR Register**

15		14		13		12		11		10		9		8	
NACKMOD		FREE		STT		RESERVED		STP		MST		TRX		XA	
R/W-0h		R/W-0h		R/W-0h		R-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7		6		5		4		3		2		1		0	
RM		DLB		IRS		STB		FDF		BC					
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h			

**Table 24-19. I2CMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	NACKMOD	R/W	0h	<p>NACK mode bit. This bit is only applicable when the I2C module is acting as a receiver.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = In the slave-receiver mode: The I2C module sends an acknowledge (ACK) bit to the transmitter during each acknowledge cycle on the bus. The I2C module only sends a no-acknowledge (NACK) bit if you set the NACKMOD bit.</p> <p>In the master-receiver mode: The I2C module sends an ACK bit during each acknowledge cycle until the internal data counter counts down to 0. At that point, the I2C module sends a NACK bit to the transmitter. To have a NACK bit sent earlier, you must set the NACKMOD bit</p> <p>1h (R/W) = In either slave-receiver or master-receiver mode: The I2C module sends a NACK bit to the transmitter during the next acknowledge cycle on the bus. Once the NACK bit has been sent, NACKMOD is cleared.</p> <p>Important: To send a NACK bit in the next acknowledge cycle, you must set NACKMOD before the rising edge of the last data bit.</p>
14	FREE	R/W	0h	<p>This bit controls the action taken by the I2C module when a debugger breakpoint is encountered.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = When I2C module is master: If SCL is low when the breakpoint occurs, the I2C module stops immediately and keeps driving SCL low, whether the I2C module is the transmitter or the receiver. If SCL is high, the I2C module waits until SCL becomes low and then stops.</p> <p>When I2C module is slave: A breakpoint forces the I2C module to stop when the current transmission/reception is complete.</p> <p>1h (R/W) = The I2C module runs free that is, it continues to operate when a breakpoint occurs.</p>
13	STT	R/W	0h	<p>START condition bit (only applicable when the I2C module is a master). The RM, STT, and STP bits determine when the I2C module starts and stops data transmissions (see Table 9-6). Note that the STT and STP bits can be used to terminate the repeat mode, and that this bit is not writable when IRS = 0.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = In the master mode, STT is automatically cleared after the START condition has been generated.</p> <p>1h (R/W) = In the master mode, setting STT to 1 causes the I2C module to generate a START condition on the I2C-bus</p>
12	RESERVED	R	0h	Reserved

**Table 24-19. I2CMDR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	STP	R/W	0h	<p>STOP condition bit (only applicable when the I2C module is a master).</p> <p>In the master mode, the RM,STT, and STP bits determine when the I2C module starts and stops data transmissions.</p> <p>Note that the STT and STP bits can be used to terminate the repeat mode, and that this bit is not writable when IRS=0. When in non-repeat mode, at least one byte must be transferred before a stop condition can be generated. The I2C module delays clearing of this bit until after the I2CSTR[SCD] bit is set. To avoid disrupting the I2C state machine, the user must wait until this bit is clear before initiating a new message.</p> <p>Reset type: SYSRSn            0h (R/W) = STP is automatically cleared after the STOP condition has been generated            1h (R/W) = STP has been set by the device to generate a STOP condition when the internal data counter of the I2C module counts down to 0.</p>
10	MST	R/W	0h	<p>Master mode bit.</p> <p>MST determines whether the I2C module is in the slave mode or the master mode. MST is automatically changed from 1 to 0 when the I2C master generates a STOP condition</p> <p>Reset type: SYSRSn            0h (R/W) = Slave mode. The I2C module is a slave and receives the serial clock from the master.            1h (R/W) = Master mode. The I2C module is a master and generates the serial clock on the SCL pin.</p>
9	TRX	R/W	0h	<p>Transmitter mode bit.</p> <p>When relevant, TRX selects whether the I2C module is in the transmitter mode or the receiver mode.</p> <p>Reset type: SYSRSn            0h (R/W) = Receiver mode. The I2C module is a receiver and receives data on the SDA pin.            1h (R/W) = Transmitter mode. The I2C module is a transmitter and transmits data on the SDA pin.</p>
8	XA	R/W	0h	<p>Expanded address enable bit.</p> <p>Reset type: SYSRSn            0h (R/W) = 7-bit addressing mode (normal address mode). The I2C module transmits 7-bit slave addresses (from bits 6-0 of I2CSAR), and its own slave address has 7 bits (bits 6-0 of I2COAR).            1h (R/W) = 10-bit addressing mode (expanded address mode). The I2C module transmits 10-bit slave addresses (from bits 9-0 of I2CSAR), and its own slave address has 10 bits (bits 9-0 of I2COAR).</p>
7	RM	R/W	0h	<p>Repeat mode bit (only applicable when the I2C module is a master-transmitter).</p> <p>The RM, STT, and STP bits determine when the I2C module starts and stops data transmissions</p> <p>Reset type: SYSRSn            0h (R/W) = Nonrepeat mode. The value in the data count register (I2CCNT) determines how many bytes are received/transmitted by the I2C module.            1h (R/W) = Repeat mode. A data byte is transmitted each time the I2CDXR register is written to (or until the transmit FIFO is empty when in FIFO mode) until the STP bit is manually set. The value of I2CCNT is ignored. The ARDY bit/interrupt can be used to determine when the I2CDXR (or FIFO) is ready for more data, or when the data has all been sent and the CPU is allowed to write to the STP bit.</p>

**Table 24-19. I2CMDR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	DLB	R/W	0h	<p>Digital loopback mode bit.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Digital loopback mode is disabled.</p> <p>1h (R/W) = Digital loopback mode is enabled. For proper operation in this mode, the MST bit must be 1.</p> <p>In the digital loopback mode, data transmitted out of I2CDXR is received in I2CDRR after n device cycles by an internal path, where:</p> <p><math>n = ((I2C \text{ input clock frequency} / \text{module clock frequency}) \times 8)</math></p> <p>The transmit clock is also the receive clock. The address transmitted on the SDA pin is the address in I2COAR.</p> <p>Note: The free data format (FDF = 1) is not supported in the digital loopback mode.</p>
5	IRS	R/W	0h	<p>I2C module reset bit.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The I2C module is in reset/disabled. When this bit is cleared to 0, all status bits (in I2CSTR) are set to their default values.</p> <p>1h (R/W) = The I2C module is enabled. This has the effect of releasing the I2C bus if the I2C peripheral is holding it.</p>
4	STB	R/W	0h	<p>START byte mode bit. This bit is only applicable when the I2C module is a master. As described in version 2.1 of the Philips Semiconductors I2C-bus specification, the START byte can be used to help a slave that needs extra time to detect a START condition. When the I2C module is a slave, it ignores a START byte from a master, regardless of the value of the STB bit.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The I2C module is not in the START byte mode.</p> <p>1h (R/W) = The I2C module is in the START byte mode. When you set the START condition bit (STT), the I2C module begins the transfer with more than just a START condition. Specifically, it generates:</p> <ol style="list-style-type: none"> <li>1. A START condition</li> <li>2. A START byte (0000 0001b)</li> <li>3. A dummy acknowledge clock pulse</li> <li>4. A repeated START condition</li> </ol> <p>Then, as normal, the I2C module sends the slave address that is in I2CSAR.</p>
3	FDF	R/W	0h	<p>Free data format mode bit.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit.</p> <p>1h (R/W) = Free data format mode is enabled. Transfers have the free data (no address) format described in Section 9.2.5.</p> <p>The free data format is not supported in the digital loopback mode (DLB=1).</p>

**Table 24-19. I2CMDR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2-0	BC	R/W	0h	<p>Bit count bits.</p> <p>BC defines the number of bits (1 to 8) in the next data byte that is to be received or transmitted by the I2C module. The number of bits selected with BC must match the data size of the other device. Notice that when BC = 000b, a data byte has 8 bits. BC does not affect address bytes, which always have 8 bits.</p> <p>Note: If the bit count is less than 8, receive data is right-justified in I2CDRR(7-0), and the other bits of I2CDRR(7-0) are undefined. Also, transmit data written to I2CDXR must be right-justified</p> <p>Reset type: SYSRStn</p> <p>0h (R/W) = 8 bits per data byte            1h (R/W) = 1 bit per data byte            2h (R/W) = 2 bits per data byte            3h (R/W) = 3 bits per data byte            4h (R/W) = 4 bits per data byte            5h (R/W) = 5 bits per data byte            6h (R/W) = 6 bits per data byte            7h (R/W) = 7 bits per data byte</p>

**24.6.2.11 I2CISRC Register (Offset = Ah) [reset = 0h]**

I2CISRC is shown in [Figure 24-28](#) and described in [Table 24-20](#).

Return to the [Summary Table](#).

The I2C interrupt source register (I2CISRC) is a 16-bit register used by the CPU to determine which event generated the I2C interrupt.

**Figure 24-28. I2CISRC Register**

15	14	13	12	11	10	9	8
RESERVED				WRITE_ZEROS			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED				INTCODE			
R-0h				R-0h			

**Table 24-20. I2CISRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-8	WRITE_ZEROS	R/W	0h	TI internal testing bits These reserved bit locations should always be written as zeros. Reset type: SYSRSn
7-3	RESERVED	R	0h	Reserved
2-0	INTCODE	R	0h	Interrupt code bits. The binary code in INTCODE indicates the event that generated an I2C interrupt.  A CPU read will clear this field. If another lower priority interrupt is pending and enabled, the value corresponding to that interrupt will then be loaded. Otherwise, the value will stay cleared.  In the case of an arbitration lost, a no-acknowledgment condition detected, or a stop condition detected, a CPU read will also clear the associated interrupt flag bit in the I2CSTR register.  Emulator reads will not affect the state of this field or of the status bits in the I2CSTR register. Reset type: SYSRSn 0h (R/W) = None 1h (R/W) = Arbitration lost 2h (R/W) = No-acknowledgment condition detected 3h (R/W) = Registers ready to be accessed 4h (R/W) = Receive data ready 5h (R/W) = Transmit data ready 6h (R/W) = Stop condition detected 7h (R/W) = Addressed as slave



**24.6.2.12 I2CEMDR Register (Offset = Bh) [reset = 1h]**

I2CEMDR is shown in [Figure 24-29](#) and described in [Table 24-21](#).

Return to the [Summary Table](#).

I2C Extended Mode

**Figure 24-29. I2CEMDR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						FCM	BC
R-0h						R/W-0h	R/W-1h

**Table 24-21. I2CEMDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1	FCM	R/W	0h	<p>Forward Compatibility mode.</p> <p>This bit when programmed brings the functionality of Tx request only when Tx data required regardless of data status in Tx buffer for non-FIFO mode.</p> <p>This register affects the XRDY behavior hence needs to be set after releasing the IRS (I2CEMDR[5]).</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Legacy functionality of requesting Tx data upon buffer copy to shift register or upon start condition is active. Stale data is reused after illegal start, ARB Lost, NACK conditions.</p> <p>1h (R/W) = New functionality of requesting data only upon ACK (address/data) is active.</p>
0	BC	R/W	1h	<p>Backwards compatibility mode.</p> <p>This bit affects the timing of the transmit status bits (XRDY and XSMT) in the I2CSTR register when in slave transmitter mode.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = See the "Backwards Compatibility Mode Bit, Slave Transmitter" Figure for details.</p> <p>1h (R/W) = See the "Backwards Compatibility Mode Bit, Slave Transmitter" Figure for details.</p>

**24.6.2.13 I2CPSC Register (Offset = Ch) [reset = 0h]**

I2CPSC is shown in [Figure 24-30](#) and described in [Table 24-22](#).

Return to the [Summary Table](#).

The I2C prescaler register (I2CPSC) is a 16-bit register (see [Figure 14-21](#)) used for dividing down the I2C input clock to obtain the desired module clock for the operation of the I2C module. See the device-specific data manual for the supported range of values for the module clock frequency.

IPSC must be initialized while the I2C module is in reset (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

**Figure 24-30. I2CPSC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
IPSC							
R/W-0h							

**Table 24-22. I2CPSC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	IPSC	R/W	0h	<p>I2C prescaler divide-down value.</p> <p>IPSC determines how much the CPU clock is divided to create the module clock of the I2C module:</p> <p>module clock frequency = I2C input clock frequency / (IPSC + 1)</p> <p>Note: IPSC must be initialized while the I2C module is in reset (IRS = 0 in I2CMDR).</p> <p>Reset type: SYSRSn</p>

#### 24.6.2.14 I2CFFTX Register (Offset = 20h) [reset = 0h]

I2CFFTX is shown in [Figure 24-31](#) and described in [Table 24-23](#).

Return to the [Summary Table](#).

The I2C transmit FIFO register (I2CFFTX) is a 16-bit register that contains the I2C FIFO mode enable bit as well as the control and status bits for the transmit FIFO mode of operation on the I2C peripheral.

**Figure 24-31. I2CFFTX Register**

15		14		13		12		11		10		9		8	
RESERVED		I2CFFEN		TXFFRST						TXFFST					
R-0h		R/W-0h		R/W-0h						R-0h					
7		6		5		4		3		2		1		0	
TXFFINT		TXFFINTCLR		TXFFIENA						TXFFIL					
R-0h		R-0/W1S-0h		R/W-0h						R/W-0h					

**Table 24-23. I2CFFTX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	I2CFFEN	R/W	0h	I2C FIFO mode enable bit. This bit must be enabled for either the transmit or the receive FIFO to operate correctly. Reset type: SYSRSn 0h (R/W) = Disable the I2C FIFO mode. 1h (R/W) = Enable the I2C FIFO mode.
13	TXFFRST	R/W	0h	Transmit FIFO Reset Reset type: SYSRSn 0h (R/W) = Reset the transmit FIFO pointer to 0000 and hold the transmit FIFO in the reset state. 1h (R/W) = Enable the transmit FIFO operation.
12-8	TXFFST	R	0h	Contains the status of the transmit FIFO: xxxxx Transmit FIFO contains xxxxx bytes. 00000 Transmit FIFO is empty.  Note: Since these bits are reset to zero, the transmit FIFO interrupt flag will be set when the transmit FIFO operation is enabled and the I2C is taken out of reset. This will generate a transmit FIFO interrupt if enabled. To avoid any detrimental effects from this, write a one to the TXFFINTCLR once the transmit FIFO operation is enabled and the I2C is taken out of reset. Reset type: SYSRSn
7	TXFFINT	R	0h	Transmit FIFO interrupt flag. This bit cleared by a CPU write of a 1 to the TXFFINTCLR bit. If the TXFFIENA bit is set, this bit will generate an interrupt when it is set. Reset type: SYSRSn 0h (R/W) = Transmit FIFO interrupt condition has not occurred. 1h (R/W) = Transmit FIFO interrupt condition has occurred.
6	TXFFINTCLR	R-0/W1S	0h	Transmit FIFO Interrupt Flag Clear Reset type: SYSRSn 0h (R/W) = Writes of zeros have no effect. Reads return a 0. 1h (R/W) = Writing a 1 to this bit clears the TXFFINT flag.
5	TXFFIENA	R/W	0h	Transmit FIFO Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disabled. TXFFINT flag does not generate an interrupt when set. 1h (R/W) = Enabled. TXFFINT flag does generate an interrupt when set.

**Table 24-23. I2CFFTX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	TXFFIL	R/W	0h	<p>Transmit FIFO interrupt level.</p> <p>These bits set the status level that will set the transmit interrupt flag. When the TXFFST4-0 bits reach a value equal to or less than these bits, the TXFFINT flag will be set. This will generate an interrupt if the TXFFIENA bit is set. Because the I2C on this device has a 16-level transmit FIFO, these bits cannot be configured for an interrupt of more than 16 FIFO levels.</p> <p>Reset type: SYSRSn</p>

**24.6.2.15 I2CFFRX Register (Offset = 21h) [reset = 0h]**

I2CFFRX is shown in [Figure 24-32](#) and described in [Table 24-24](#).

Return to the [Summary Table](#).

The I2C receive FIFO register (I2CFFRX) is a 16-bit register that contains the control and status bits for the receive FIFO mode of operation on the I2C peripheral.

**Figure 24-32. I2CFFRX Register**

15	14	13	12	11	10	9	8
RESERVED		RXFFRST					RXFFST
R-0h		R/W-0h					R-0h
7	6	5	4	3	2	1	0
RXFFINT	RXFFINTCLR	RXFFIENA				RXFFIL	
R-0h	R-0/W1S-0h	R/W-0h				R/W-0h	

**Table 24-24. I2CFFRX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0h	Reserved
13	RXFFRST	R/W	0h	I2C receive FIFO reset bit Reset type: SYSRSn 0h (R/W) = Reset the receive FIFO pointer to 0000 and hold the receive FIFO in the reset state. 1h (R/W) = Enable the receive FIFO operation.
12-8	RXFFST	R	0h	Contains the status of the receive FIFO: xxxxx Receive FIFO contains xxxxx bytes 00000 Receive FIFO is empty. Reset type: SYSRSn
7	RXFFINT	R	0h	Receive FIFO interrupt flag. This bit cleared by a CPU write of a 1 to the RXFFINTCLR bit. If the RXFFIENA bit is set, this bit will generate an interrupt when it is set Reset type: SYSRSn 0h (R/W) = Receive FIFO interrupt condition has not occurred. 1h (R/W) = Receive FIFO interrupt condition has occurred.
6	RXFFINTCLR	R-0/W1S	0h	Receive FIFO interrupt flag clear bit. Reset type: SYSRSn 0h (R/W) = Writes of zeros have no effect. Reads return a zero. 1h (R/W) = Writing a 1 to this bit clears the RXFFINT flag.
5	RXFFIENA	R/W	0h	Receive FIFO interrupt enable bit. Reset type: SYSRSn 0h (R/W) = Disabled. RXFFINT flag does not generate an interrupt when set. 1h (R/W) = Enabled. RXFFINT flag does generate an interrupt when set.

**Table 24-24. I2CFFRX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	RXFFIL	R/W	0h	<p>Receive FIFO interrupt level.</p> <p>These bits set the status level that will set the receive interrupt flag. When the RXFFST4-0 bits reach a value equal to or greater than these bits, the RXFFINT flag is set. This will generate an interrupt if the RXFFIENA bit is set.</p> <p>Note: Since these bits are reset to zero, the receive FIFO interrupt flag will be set if the receive FIFO operation is enabled and the I2C is taken out of reset. This will generate a receive FIFO interrupt if enabled. To avoid this, modify these bits on the same instruction as or prior to setting the RXFFRST bit. Because the I2C on this device has a 16-level receive FIFO, these bits cannot be configured for an interrupt of more than 16 FIFO levels.</p> <p>Reset type: SYSRSn</p>

### 24.6.3 Register to Driverlib Function Mapping

**Table 24-25. I2C Registers to Driverlib Functions**

File	Driverlib Function
<b>OAR</b>	
i2c.h	I2C_setOwnSlaveAddress
<b>IER</b>	
i2c.c	I2C_enableInterrupt
i2c.c	I2C_disableInterrupt
<b>STR</b>	
i2c.c	I2C_getInterruptStatus
i2c.c	I2C_clearInterruptStatus
i2c.h	I2C_isBusBusy
i2c.h	I2C_getStatus
i2c.h	I2C_clearStatus
<b>CLKL</b>	
i2c.c	I2C_initMaster
<b>CLKH</b>	
i2c.c	I2C_initMaster
<b>CNT</b>	
i2c.h	I2C_setDataCount
<b>DRR</b>	
i2c.h	I2C_getData
<b>SAR</b>	
i2c.h	I2C_setSlaveAddress
<b>DXR</b>	
i2c.h	I2C_putData
<b>MDR</b>	
i2c.h	I2C_enableModule
i2c.h	I2C_disableModule
i2c.h	I2C_setConfig
i2c.h	I2C_setBitCount
i2c.h	I2C_sendStartCondition
i2c.h	I2C_sendStopCondition
i2c.h	I2C_sendNACK
i2c.h	I2C_getStopConditionStatus
i2c.h	I2C_setAddressMode
i2c.h	I2C_setEmulationMode
i2c.h	I2C_enableLoopback
i2c.h	I2C_disableLoopback
<b>ISRC</b>	
i2c.h	I2C_getInterruptSource
<b>PSC</b>	
i2c.c	I2C_initMaster
<b>FFTX</b>	
i2c.c	I2C_enableInterrupt
i2c.c	I2C_disableInterrupt
i2c.c	I2C_getInterruptStatus
i2c.c	I2C_clearInterruptStatus
i2c.h	I2C_enableFIFO

**Table 24-25. I2C Registers to Driverlib Functions (continued)**

<b>File</b>	<b>Driverlib Function</b>
i2c.h	I2C_disableFIFO
i2c.h	I2C_setFIFOInterruptLevel
i2c.h	I2C_getFIFOInterruptLevel
i2c.h	I2C_getTxFIFOStatus
<b>FFRX</b>	
i2c.c	I2C_enableInterrupt
i2c.c	I2C_disableInterrupt
i2c.c	I2C_getInterruptStatus
i2c.c	I2C_clearInterruptStatus
i2c.h	I2C_enableFIFO
i2c.h	I2C_disableFIFO
i2c.h	I2C_setFIFOInterruptLevel
i2c.h	I2C_getFIFOInterruptLevel
i2c.h	I2C_getRxFIFOStatus



---

---

## ***Power Management Bus Module (PMBus)***

---

---

This chapter describes the features and operation of the Power Management Bus (PMBus) module.

<b>Topic</b>	<b>Page</b>
<b>25.1 Introduction .....</b>	<b>2330</b>
<b>25.2 Configuring Device Pins .....</b>	<b>2331</b>
<b>25.3 Slave Mode Operation .....</b>	<b>2331</b>
<b>25.4 Master Mode Operation .....</b>	<b>2340</b>
<b>25.5 PMBus Registers .....</b>	<b>2349</b>

## 25.1 Introduction

The PMBus module provides an interface between the microcontroller and devices compliant with the SMI Forum PMBus Specification Part I version 1.0 and Part II version 1.1. PMBus is based on SMBus, which uses a similar physical layer to I2C. This guide assumes the reader is familiar with the PMBus, SMBus, and I2C bus specifications.

### 25.1.1 Features

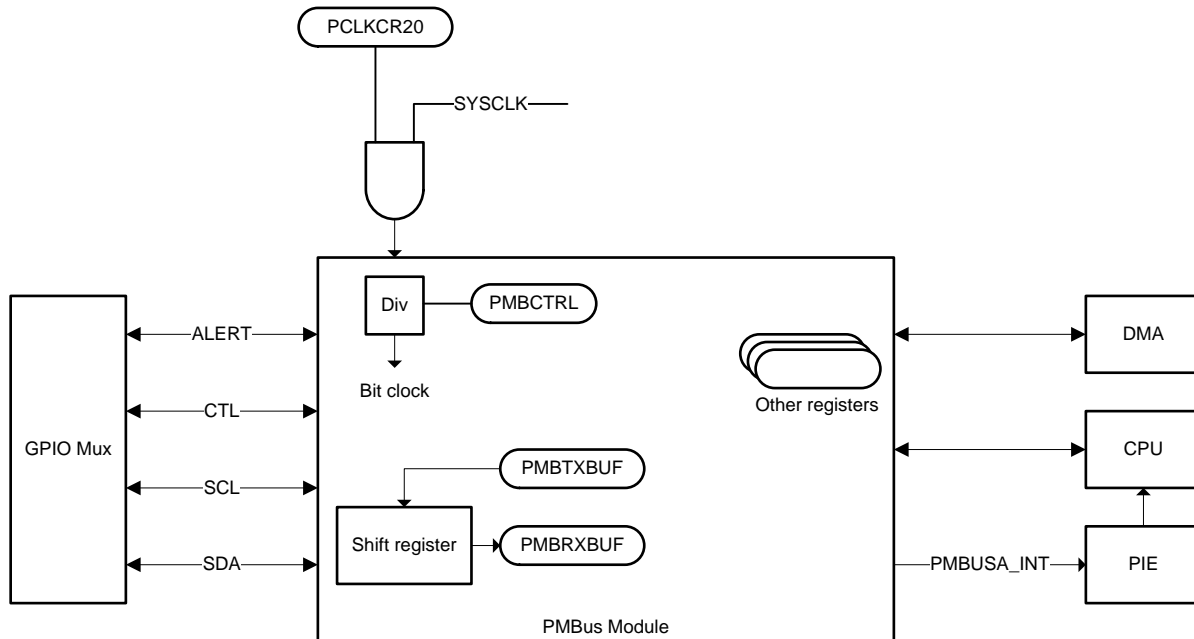
The PMBus module has the following features:

- Compliance with the SMI Forum PMBus Specification (Part I v1.0 and Part II v1.1)
- Support for master and slave modes
- Support for I2C modes
- Support for three speeds:
  - Standard Mode: Up to 100 kHz
  - Fast Mode: Up to 400 kHz
- Packet error checking
- CONTROL and ALERT signals
- Clock high and low time-outs
- Four-byte transmit and receive buffers
- One maskable interrupt, which can be generated by several conditions:
  - Receive data ready
  - Transmit buffer empty
  - Slave address received
  - End of message
  - ALERT input asserted
  - Clock low time-out
  - Clock high time-out
  - Bus free

### 25.1.2 Functional Description

The following figure shows the block diagram for PMBus.

Figure 25-1. PMBus Module Conceptual Block Diagram



The PMBus module handles the lower levels of the PMBus protocol. In addition to controlling signal levels and timing, parsing addresses, and buffering data, it also directly supports complex transactions such as Read Word and Process Call.

There are four PMBus signals: SCL, SDA, CONTROL, and ALERT.

- SCL is the bus clock. It is normally controlled by the master, but may be held low by a slave to delay a transaction and allow more time for processing.
- SDA is the bidirectional data line.
- CONTROL is a slave input that can trigger an interrupt. It can be used to tell a slave device to shut down.
- ALERT is a slave output/master input that allows a slave to request attention from the master.

The SDA and SCL timings produced by the module are derived from SYSCLK. To comply with the PMBus timing specs, the bit clock divider must be set via the PMBTIMCLK register to provide a bit clock of 10 MHz or less.

## 25.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification is set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pullups are configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

## 25.3 Slave Mode Operation

This section describes the configuration and operation of the PMBus module in slave mode.

### 25.3.1 Configuration

To configure the module, write a suitable clock divider to the PMBCTRL register's CLKDIV field to produce a bit clock frequency of less than 10 MHz. To activate slave mode, set the SLAVE\_EN bit in the PMBCTRL register. Next, set up the PMBSC register. The following options are configurable:

- Slave address and mask (SLAVE\_ADDR and SLAVE\_MASK): Sets the slave address and mask for message acceptance.
- Manual slave address acknowledgement (MAN\_SLAVE\_ACK): When enabled, allows software to decide whether to acknowledge (ACK) an address. When disabled, the decision to ACK is made automatically based on the slave address and mask.
- PEC enable (PEC\_ENA): Set this bit if Packet Error Checking (PEC) is used on the bus.
- Manual command byte acknowledgement (MAN\_CMD): Similar to manual slave acknowledgement, setting this bit allows software to decide whether to acknowledge (ACK) a command byte.
- Number of bytes to acknowledge automatically (RX\_BYTE\_ACK\_CNT): This is normally set to the max value, which allows the entire receive buffer to be used. However, smaller values may be used if the application requires that erroneous messages be detected and not acknowledged (NACKed) as soon as possible.

Manual acknowledgement is done by writing a one to the PMBACK register. Even with automatic acknowledgement, some writes to PMBACK are required. If the message (not including the address) is longer than 4 bytes, each packet of 4 bytes must be acknowledged. The PMBus module will stretch the clock (hold it low) until an ACK is issued. The module will then pull the data line low and release the clock, providing the ACK signal to the master.

If the complete message or the last part of the message is less than 4 bytes (or the RX\_BYTE\_ACK\_CNT limit), do not write to PMBACK.

Writing a zero to the PMBACK bit will send a NACK. This may only be done when the module is waiting for an acknowledgement. If a zero is written at any other time, the NACK will be issued during the next message.

### 25.3.2 Message Handling

This section describes some of the message types for PMBus and how to determine which message type is being received in slave mode. It is oriented toward the most efficient mode of operation – with automatic address and command acknowledgment. It is also oriented toward having Packet Error Checking (PEC) enabled.

If automatic address acknowledgement is disabled, all messages will start by setting the SLAVE\_ADDR\_READY register bit high. Read commands will have two instructions one for the read, and one for the write. If automatic command acknowledgement is enabled, the DATA READY bit will be set high, as well. If the message has no PEC, the number of bytes available will be n-1. For example, with PEC, a QUICK COMMAND will have one byte. With no PEC, a QUICK COMMAND will have zero bytes.

Note that the byte count does not increment as bytes arrive. No bits are set in the PMBST register until a stop message is received, the receive buffer is full, or a fault occurs. Then, all appropriate bit values are placed in the register together. All that is necessary to receive a quick command is to ACK the message by writing a one to the PMBACK register.

#### 25.3.2.1 Quick Command

Quick Commands received by the PMBus module in slave mode require a simple acknowledgment of the received device address. In automatic address acknowledge mode, the module processes the quick command without firmware interaction. Upon receipt of the end of message, the firmware has the option to read the received address in the PMBHS register. In manual address acknowledge mode, the address is acknowledged by writing to the PMBACK register.

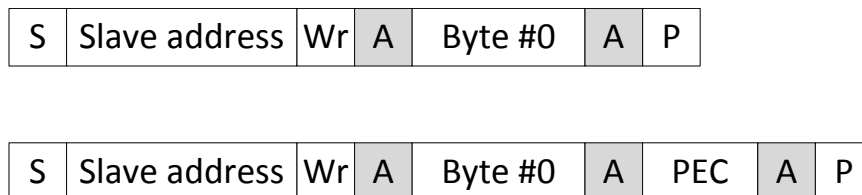
**Figure 25-2. Quick Command Message**



### 25.3.2.2 Send Byte

A Send Byte message consists of the device address, a single data byte and an optional PEC byte. To process the PEC byte correctly, PEC processing must be enabled in the PMBSC register. In automatic address acknowledge mode, the data and optional PEC byte are acknowledged without firmware interaction. The module generates an End of Message interrupt, reads the status register and finds the data ready indication bit set. In manual mode, the address is acknowledged by the firmware, while the remaining data and PEC bytes are acknowledged by the module

**Figure 25-3. Send Byte Message with and without PEC**



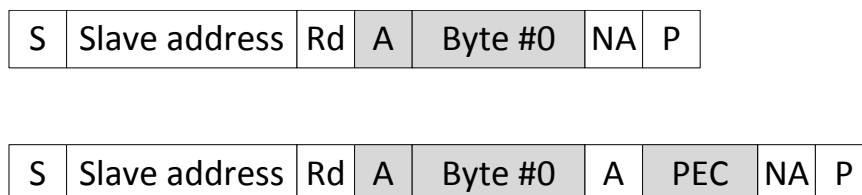
The PMBus module stores Data Byte #0 into the PMBRXBUF register. The data byte will be stored into bits 7-0. In non-PEC mode, the RX Byte Count in the PMBSTS register will indicate one byte received. If PEC processing is enabled, the PEC byte is also stored into the PMBRXBUF register, with the PEC byte residing in bits 15-8. The RX Byte Count in the PMBSTS register will indicate two bytes received. The PEC Valid bit in the PMBSTS register indicates the validity of the received PEC byte.

When a Send Byte message is received, the Data Ready bit is set along with the EOM and, assuming that the PEC is valid, the PEC valid bit. The read byte count (RD\_BYTE\_COUNT) register will contain a 2. All that is necessary to receive a send byte command is to ACK the message by writing a 1 to the PMBACK register. Before doing the ACK, read the byte from the lowest byte of the PMBRXBUF register.

### 25.3.2.3 Receive Byte

A Receive Byte message consists of the device address, a single data byte and an optional PEC byte. In automatic address acknowledge mode, the firmware receives a data request interrupt following reception of the slave address. The data byte to be sent to the master is stored into bits 7-0 of the PMBTXBUF register and Transmit Byte Count bits within the PMBSC register are set to a value of one. If PEC processing is enabled, the Transmit PEC bit (bit 19) within the PMBSC register is set to '1', along with the Enable PEC bit (bit 15). The module will automatically append the calculated PEC byte at the completion of the message.

**Figure 25-4. Receive Byte Message with and without PEC**



### 25.3.2.4 Write Byte/Word

The Write Byte and Write Word messages consist of a slave address, a command word, transmitted data bytes and an optional PEC byte. In automatic address acknowledge mode, the data bytes and optional PEC byte are acknowledged without firmware interaction. The acknowledgment of the command word is configured through the PMBSC register. The firmware receives an End of Message interrupt in all cases except for Write Word with PEC message, reads the status register and finds the data ready indication bit set.

**Figure 25-5. Write Byte and Write Word Messages with and without PEC**

S	Slave address	Wr	A	Command	A	Byte #0	A	P
---	---------------	----	---	---------	---	---------	---	---

S	Slave address	Wr	A	Command	A	Byte #0	A	PEC	A	P
---	---------------	----	---	---------	---	---------	---	-----	---	---

S	Slave address	Wr	A	Command	A	Byte #0	A	Byte #1	A	P
---	---------------	----	---	---------	---	---------	---	---------	---	---

S	Slave address	Wr	A	Command	A	Byte #0	A	Byte #1	A	PEC	A	P
---	---------------	----	---	---------	---	---------	---	---------	---	-----	---	---

In the case of a Write Word with PEC byte message, the data ready interrupt is enabled after receiving 4 bytes (command byte, the 2 data bytes and the PEC byte). The firmware reads the data from the PMBRXBUF register and must write the PMBACK register to acknowledge back to the master. The PMBus module holds SCL low until the firmware responds to the received data.

In all other cases, the EOM interrupt is received and data can be read from the PMBRXBUF register. The firmware is not required to send an acknowledgement back to the master.

The Write Byte message will look exactly the same as the Send Byte, except the RD\_BYTE\_COUNT register will contain a 3. The Write Word message will have a RD\_BYTE\_COUNT of 4.

### 25.3.2.5 Read Byte/Word

The Read Byte and Read Word messages consist of a slave address, a command word, received data bytes from a slave, and an optional PEC byte. Address and command acknowledgment is configured through the PMBSC register. In automatic mode, the PMBus module provides a data ready and data request interrupt following receipt of a repeated start and slave address. The received command byte is found in bits 7-0 of the PMBRXBUF register. The firmware responds to the data request by programming the data bytes into the PMBTXBUF register and the TX Byte Count bits in the PMBSC register. If PEC processing is enabled, the Transmit PEC bit should also be asserted. An EOM interrupt indicates completion of the message to the Master.

**Figure 25-6. Read Byte and Read Word Messages with and without PEC**

S	Slave address	Wr	A	Command	A	Sr	Slave address	Rd	A
---	---------------	----	---	---------	---	----	---------------	----	---

Byte #0	NA	P
---------	----	---

S	Slave address	Wr	A	Command	A	Sr	Slave address	Rd	A
---	---------------	----	---	---------	---	----	---------------	----	---

Byte #0	A	PEC	NA	P
---------	---	-----	----	---

S	Slave address	Wr	A	Command	A	Sr	Slave address	Rd	A
---	---------------	----	---	---------	---	----	---------------	----	---

Byte #0	A	Byte #1	NA	P
---------	---	---------	----	---

S	Slave address	Wr	A	Command	A	Sr	Slave address	Rd	A
---	---------------	----	---	---------	---	----	---------------	----	---

Byte #0	A	Byte #1	A	PEC	NA	P
---------	---	---------	---	-----	----	---

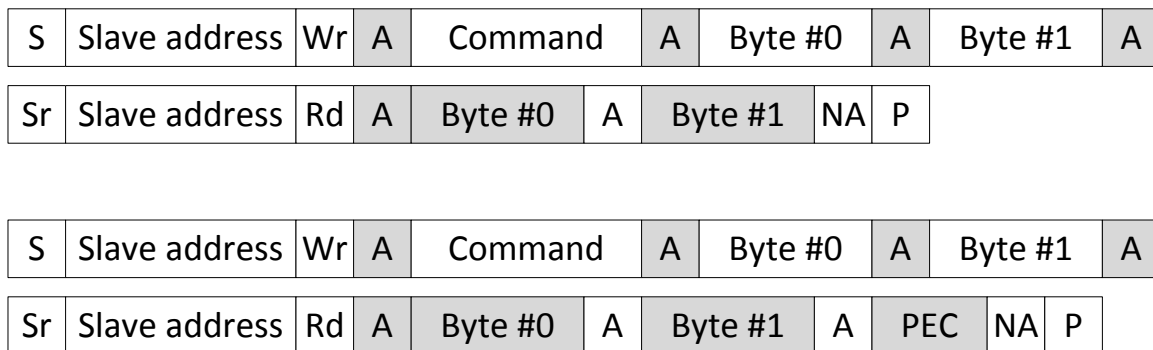
When the repeated start (Sr) signal is received, the Data Ready bit will be asserted with a RD\_BYTE\_COUNT of 1. At this point, the operation cannot be distinguished from a group command Send Byte message. When the same device address is sent out with a read, the Data Request bit will be asserted. If data has already been written to the PMTXBUF register before the Device Addr is received, the Data Request bit will not be asserted. So if group commands are also expected, it is necessary to read the Data Ready with a RD\_BYTE\_COUNT of 1, and then wait and see whether the next event is an EOM or a Data Request. If it is an EOM, the command should be processed as a group send byte. If it is a Data Request, the command should be processed as a read. Depending on the command, it could be a read byte, word, or block. If the PMBus module is polled, both the Data Ready and the Data Request bits could possibly be set between polling intervals. This should be considered in the design of the firmware.

Once the read command is recognized, you must respond by writing data to the PMBTXBUF register. It is also necessary to make sure that the values in the PMBSC register are correct. The transmit byte count and PEC bit must be set appropriately. For a read byte, the transmit byte count can be loaded with a 1. If the transmission of a PEC byte is desired, the TX\_PEC bit must be set. After this, the data can be written to PMBTXBUF, which starts the transmission. All bytes should be written to PMBTXBUF at the same time. After the master receives the message, it will NACK the last byte to indicate that the correct number of bytes have been received. This will cause the EOM bit to be set in the PMBSTS register, indicating to the firmware that the Read Byte message is complete.

### 25.3.2.6 Process Call

The Process Call protocol consists of a Write Word message, followed by a Read Word message, without a stop condition between the two messages. Address and command acknowledgement is configured through the PMBSC register. In automatic mode, following receipt of the repeated start and slave address, the PMBus module provides a data ready and a data request interrupt. The repeated start bit is set in the PMBSTS register to indicate the receipt of the first part of the Process Call message. The received command byte is found in bits 7-0 of the PMBRXBUF register, while the two data bytes received from the master can be found in bits 23-8. Upon receipt of the repeated start and a data request from the module, the firmware programs the PMBTXBUF with the 2 data bytes to be sent to the master. If PEC processing is enabled, the Transmit PEC bit within the PMBSC register is asserted. The EOM interrupt will indicate the read word portion of the Process Call message has been completed by the module

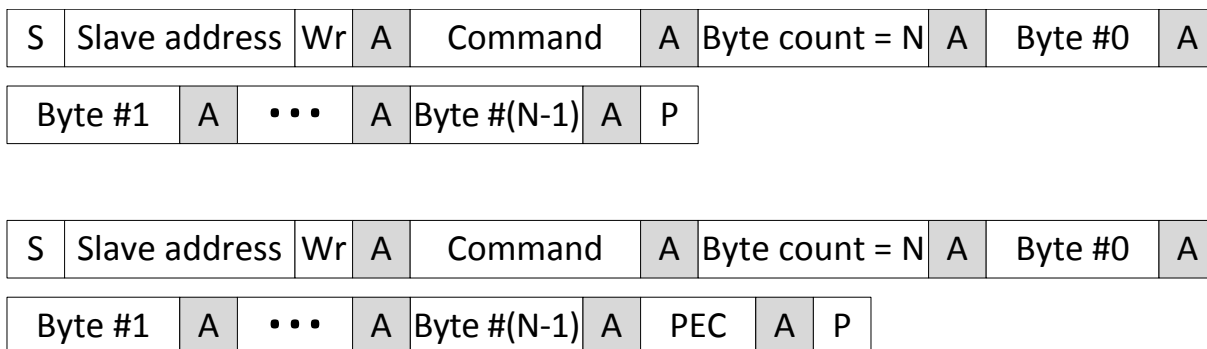
**Figure 25-7. Process Call Message with and without PEC**



### 25.3.2.7 Block Write

The Block Write protocol is similar to Write Word in its structure, except that there are more than 2 data bytes in the message. Following the receipt of the command byte, the block length and 2 data bytes, the PMBus module provides a data ready interrupt. The module waits for the firmware to read the received data and program the acknowledge register. While waiting for an ACK from the firmware, the module will drive the clock line low, stalling the bus. The data ready interrupts will continue for the duration of the message at a frequency of every 4 data bytes. The number of bytes received can be found within the PMBSTS register. At the end of the message, less than 4 bytes may be stored in the PMBRXBUF register. The PEC Valid bit can be checked to determine if the received PEC value is accurate.

**Figure 25-8. Block Write Message with and without PEC**



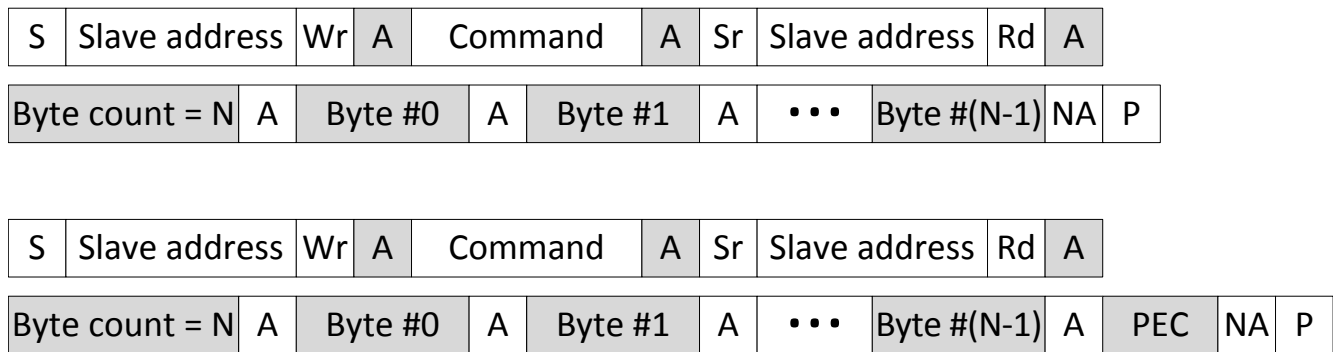


### 25.3.2.8 Block Read

The Block Read protocol is similar to a Read Word in its structure, except that there are more than 2 data bytes in the message. Following the receipt of the repeated slave address, a data ready and data request interrupt is generated by the PMBus module. The command byte received from the master can be found in bits 7-0 of the PMBRXBUF register. The SCL line is held low until the firmware programs data bytes into the PMBTXBUF register. The firmware is required to load the block length into bits 7-0 of the PMBTXBUF register during the initial programming of the register. After 4 bytes have been transmitted, the module will issue a data request interrupt and hold SCL low again until the firmware has programmed additional data into the PMBTXBUF register.

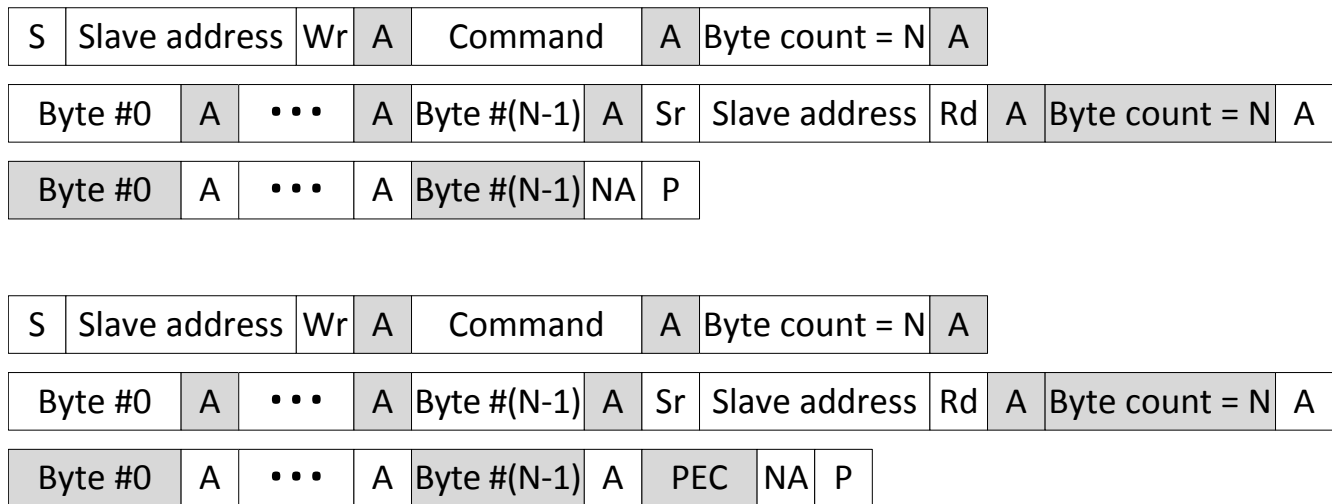
Block read starts the same as Read Word or Read Byte, but TX\_COUNT is loaded with a 4 the first time, and TX\_PEC is not set. Instead of waiting for an EOM after the first transmission, the firmware instead waits for a Data Request, indicating that the master is ready for more data. Until the last 4 or less bytes, the firmware simply writes a 4 to TX\_COUNT and then writes the 4 bytes to PMBTXBUF. TX\_PEC is left cleared. Then when the last 4 or fewer bytes are to be transmitted, the firmware writes out the appropriate byte count, sets the TX\_PEC bit, and writes the data to PMBTXBUF. The PMBus module will write out the data, followed by the PEC, and then the EOM bit will be set when the master NACKs the PEC.

**Figure 25-9. Block Read Message with and without PEC**



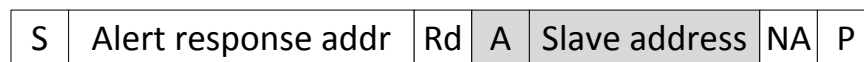
### 25.3.2.9 Block Write-Block Read Process Call

The Block Read-Block Write Process Call protocol combines the Block Write and Block Read protocols, removing the stop condition between the two messages. The processing of the Block Read-Block Write Process Call message is similar to the mode of operation for the Process Call message. After acknowledgement of the address and command bytes, the PMBus module generates a data ready interrupt upon detection of 4 data bytes or a repeated start condition. After receiving the repeated start, the firmware will be required to load transmit data to send to the master. Bits 7-0 of the initial programming of the PMBTXBUF register must represent the byte count of the block data sent to the master.

**Figure 25-10. Block Write-Block Read Process Call Message with and without PEC**


### 25.3.2.10 Alert Response

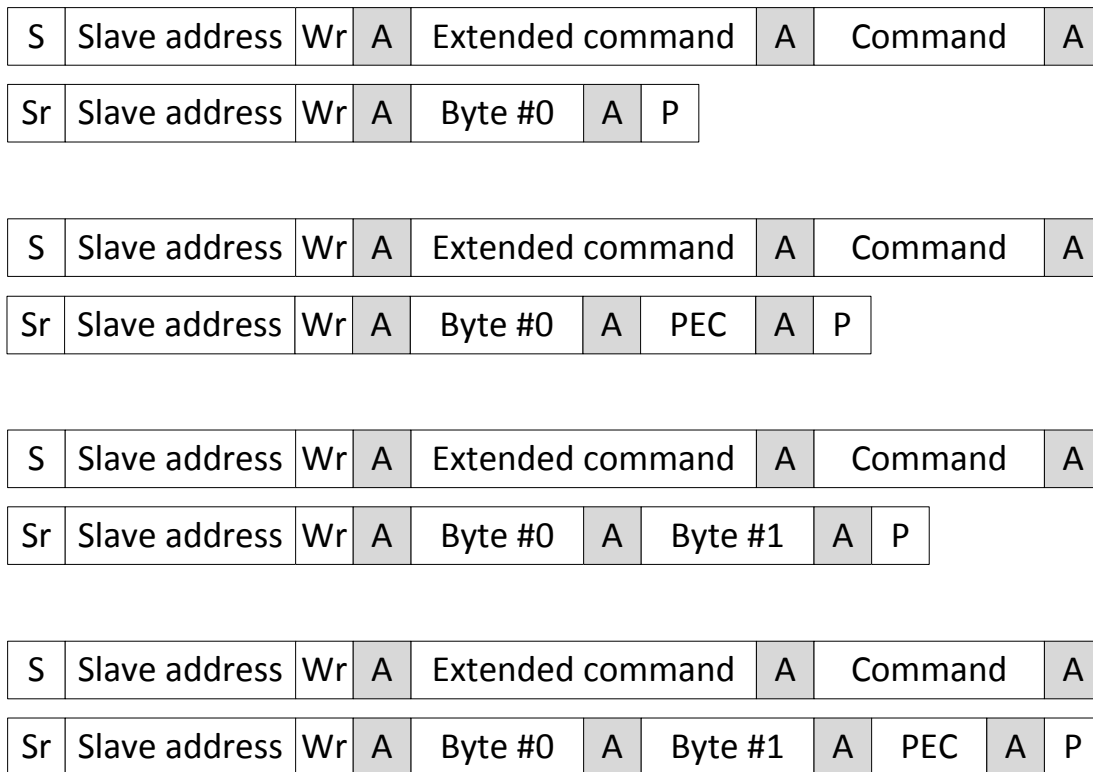
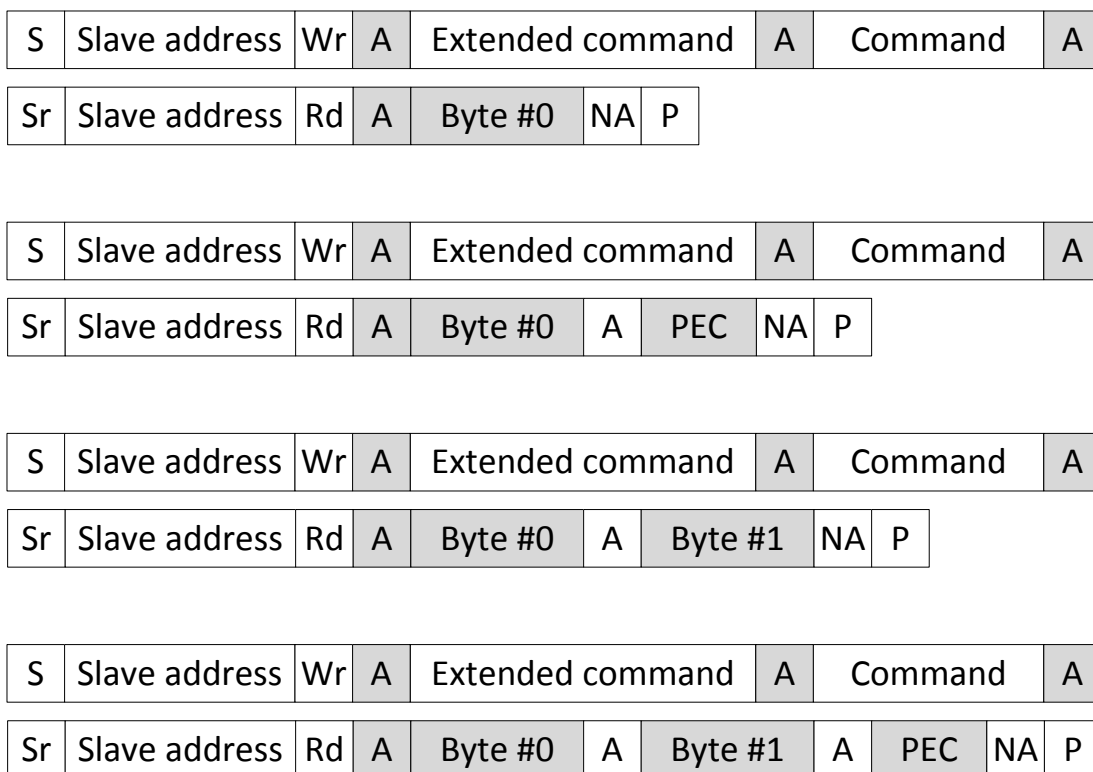
The Alert Response Message is utilized when the master detects an alert condition from a slave on the PMBus. In automatic address acknowledge mode, upon detection of the Alert Response Address, the PMBus module provides an acknowledgement to the master and sends the programmed slave address within the PMBSC register. The module only responds to the message if the Alert En bit within PMBCTRL register has been previously set. After receiving the Alert Response message, the module will clear the alert condition and enable bit within the PMBCTRL register.

**Figure 25-11. Alert Response Message**


In manual address acknowledge mode, the firmware must read the received address from the PMBRXBUF register and transmit the desired slave address back to the master. The PMBCTRL register must be reprogrammed to disable the Alert En bit used to initiate the Alert Response message from the master.

### 25.3.2.11 Extended Command

The PMBus module provides support for extended commands which allow for an extra 256 command codes. Both command bytes are stored in the PMBRXBUF register along with the data bytes. In recognizing the extended command messages, the Repeated Start bit and the Rd Byte Count Bits within the PMBSTS register are utilized. For Extended Command Write Byte/Write Word messages, the two command bytes are stored in bits 15-0 of the PMBRXBUF register. The initial command byte should hold the command extension code, representing utilization of the extended command protocol. The Repeated Start bit is also set, received after the retransmission of the device address. The Rd Byte Count equals 3 for an Ext Cmd Write Byte message and 4 for an Ext Cmd Write Word message.

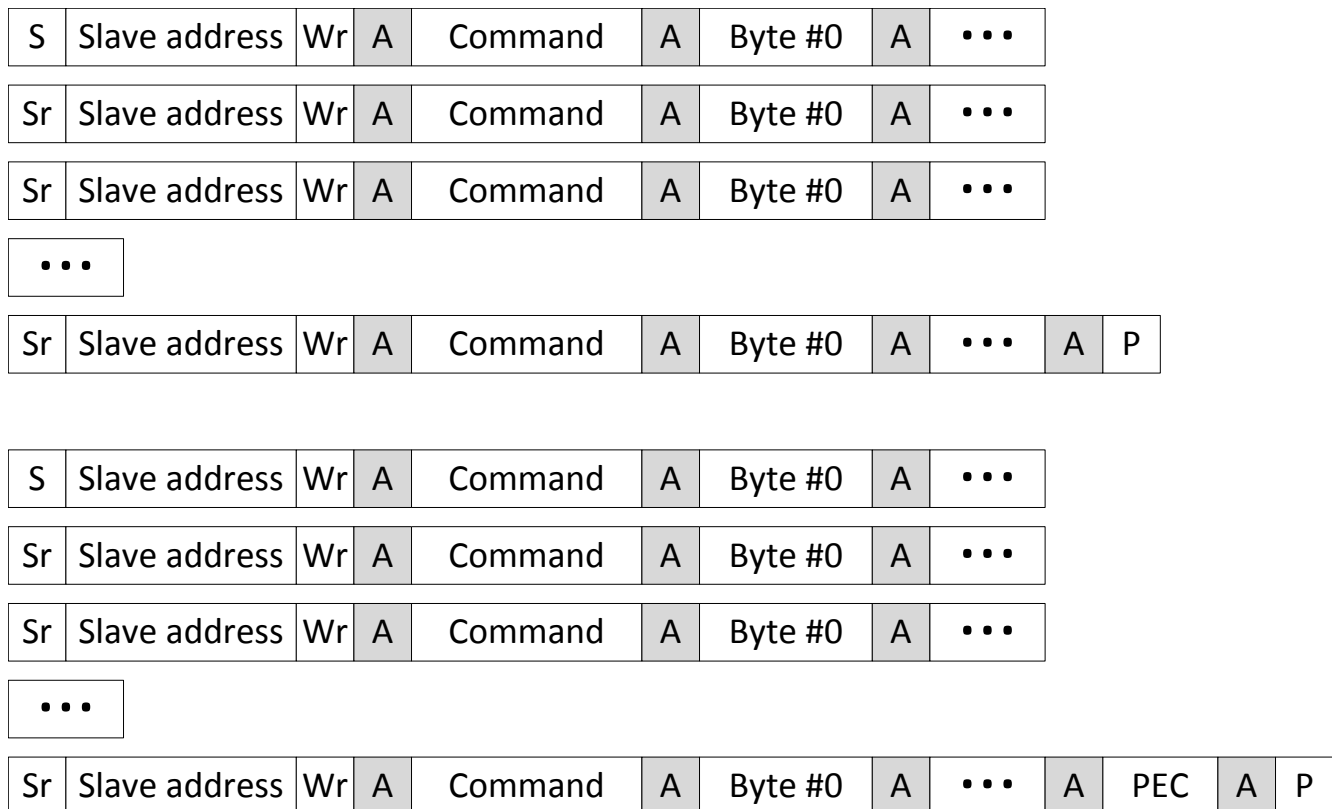
**Figure 25-12. Extended Command Write Byte and Write Word Messages with and without PEC**

**Figure 25-13. Extended Command Write Byte and Write Word Messages with and without PEC**


For the Extended Command Read Byte/Read Word messages, the module will generate a data ready and data request interrupt following reception of the repeated device address. The two command bytes will be found in bits 15-0 of the PMBRXBUF register, with the initial command byte matching the command extension code. The firmware will be required to load transmit data to complete the message back to the master.

### 25.3.2.12 Group Command

The PMBus module supports the Group Command Protocol. The Group Command Protocol is used to send commands to more than one device within the same message. When devices on the bus detect the stop condition at the conclusion of the Group Command message, the received commands are executed concurrently. Following address and command acknowledgment, the module provides a data ready interrupt upon detection of 4 data bytes or the transmission of a repeated start on the bus. The firmware should wait for the EOM interrupt before processing the received command, as required by the use of the Group Command message.

**Figure 25-14. Group Command Message with and without PEC**



For Group Commands, the data ready bit will be set as soon as the repeated start is received. The data can then be read into memory. But the data should not be acted upon until the EOM bit is set, which will occur when all of the messages have been received. Other than this delayed EOM, there is no difference for the slave firmware in receiving a Group Command than any other write message.

## 25.4 Master Mode Operation

This section describes the configuration and operation of the PMBus module in master mode.

### 25.4.1 Configuration

First, write a suitable clock divider to the PMBCTRL register's CLKDIV field to produce a bit clock frequency of less than 10 MHz. To activate master mode, set the MASTER\_EN bit and clear the SLAVE\_EN bit in the PMBCTRL register. For each transaction, set up the PMBMC register. The following options are configurable:

- Slave address (SLAVE\_ADDR): Sets the slave address for the next transaction.
- PEC enable (PEC\_ENA): If Packet Error Checking (PEC) is used on the bus, set this bit.
- Extended command code enable (EXT\_CMD): When set, uses two bytes for commands.
- Command code enable (CMD\_ENA): When set, sends a command byte at the start of the transaction.
- Byte count (BYTE\_COUNT): Determines the number of data bytes to transfer. This does not include the block length byte, which is generated automatically when needed.
- Special command enables (GRP\_CMD and PRC\_CALL): Enables special behavior for group commands and process calls.

Writing to the PMBMC register will start a transfer.

Manual acknowledgement of received data is not needed.

### 25.4.2 Message Handling

This section describes the behavior and required configuration for each command type.

#### 25.4.2.1 Quick Command

Quick commands are initiated in master mode by simply programming the desired slave device address into the PMBMC. The byte count within the PMBMC register is configured to 0 bytes by writing all zeros to bits 15-8. Upon transmission of the device address, the PMBus module will monitor the slave acknowledgement of the address. If the address is not acknowledged, the NACK bit within the status register is enabled and the PMBus module automatically sends a stop condition on the bus to terminate the message. If the address is acknowledged, a data request is issued to the processor. The firmware writes a zero to the PMBACK to terminate the message, forcing the PMBus modules to write a stop condition onto the bus.

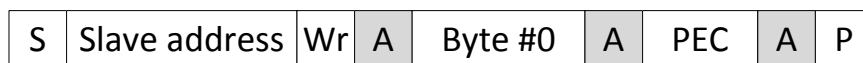
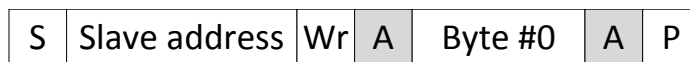
**Figure 25-15. Quick Command Message**



#### 25.4.2.2 Send Byte

A Send Byte message consists of the device address, a single data byte and an optional PEC byte. To initiate a Send Byte message, the data byte to be transmitted to the slave is loaded into bits 7-0 of the PMBTXBUF register. The PMBMC register is configured with the device address. To transmit a PEC byte with the message, the PEC\_EN bit within the PMBMC register is asserted high when the address is programmed.

**Figure 25-16. Send Byte Message with and without PEC**

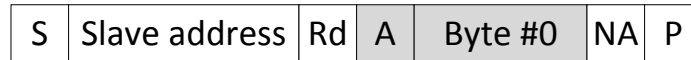


After programming the PMBMC register, the PMBus module transmits the Send Byte message. The firmware can wait for an End of Message interrupt from the PMBus module. Upon receipt of the EOM interrupt, the PMBSTS register is read to verify the slave properly acknowledged the transmitted data.

### 25.4.2.3 Receive Byte

A Receive Byte message consists of the device address, a single data byte and an optional PEC byte. Data is being read from the slave in a Receive Byte message. To initiate a Receive Byte message, the firmware programs the device address, the R/W bit and the optional PEC\_EN into the PMBMC register. The R/W bit is enabled high to indicate a read message type (data transmitted from slave to master).

**Figure 25-17. Receive Byte Message with and without PEC**

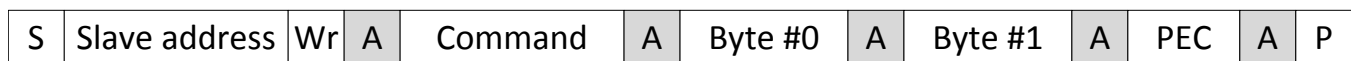
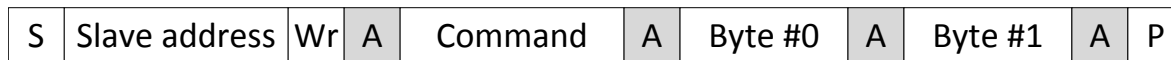


After programming the PMBMC register, the PMBus module transmits the Receive Byte message. The firmware can wait for an End of Message interrupt from the PMBus module to verify the accuracy of the message transmission. Upon receipt of the EOM interrupt, the PMBSTS register is read to verify proper slave acknowledgement of the device address and to determine if any data is available for reading in the PMBRXBUF register. If PEC\_EN was asserted in the PMBMC register, the PEC\_VALID bit in the PMBSTS register is also checked to ensure a proper PEC byte was received from the slave with the received data.

### 25.4.2.4 Write Byte/Word

The Write Byte and Write Word messages consist of a device address, a command byte, transmitted data bytes and an optional PEC byte. Write Byte messages include a single byte, while the Write Word messages support transmission of 2 bytes to the corresponding slave module. Similar to the Send Byte protocol, the PMBMC register is configured to send 1 or 2 bytes, the CMD\_EN bit is set to enable command byte transmission and the optional PEC\_EN bit is set.

**Figure 25-18. Write Byte and Write Word Messages with and without PEC**



With the command byte transmission enabled, the format of the PMBTXBUF register differs from the Send Byte protocol. In bits 7-0, the firmware must program the command byte to be sent to the slave. The data byte(s) are programmed into bits 15-8 and bits 23-16.

After programming the PMBMC register, the PMBus module transmits the Write Byte/Word message. The firmware can wait for an End of Message interrupt from the module to verify the accuracy of the message transmission. The PMBSTS register indicates if the slave acknowledged the message properly.

### 25.4.2.5 Read Byte/Word

The Read Byte and Read Word messages consist of a device address, a command byte, received data bytes from a slave and an optional PEC byte. Read Byte messages include a single byte, while the Read Word message protocol supports receipt of 2 bytes from the slave. Similar to the Receive Byte Protocol, the PMBMC register is configured to receive 1 or 2 bytes, the CMD\_EN bit is set and the PEC\_EN is configured to expect or not expect a PEC byte appended to the message. The PMBus module will automatically terminate the message after the expected number of bytes is received from the slave or if the slave does not properly acknowledge any portion of the message.

**Figure 25-19. Read Byte and Read Word Messages with and without PEC**

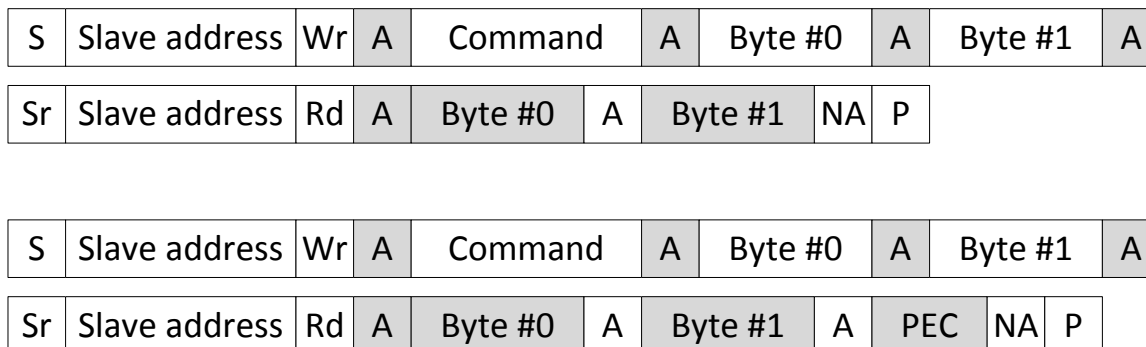


In addition to programming the PMBMC register, the firmware is expected to load the command byte into bits 7-0 of the PMBTXBUF register. Any data received from the slave will be found in the PMBRXBUF register.

### 25.4.2.6 Process Call

The Process Call protocol consists of a Write Word message, followed by a Read Word message, without a stop condition between the two messages. A PEC byte can be appended to the read data from the slave as an option to the message protocol. The PMBMC register includes a PRC\_CALL bit, which enables the transmission of a Process Call message onto the PMBus. The PMBus module will automatically generate a repeated start condition and initiate the Read Word portion of the message when the process call bit is enabled.



**Figure 25-20. Process Call Message with and without PEC**


To complete the Write Word portion of the Process Call, the PMBTXBUF register is loaded with the command byte in bits 7-0 and the data bytes are loaded into bits 23-8 of the register.

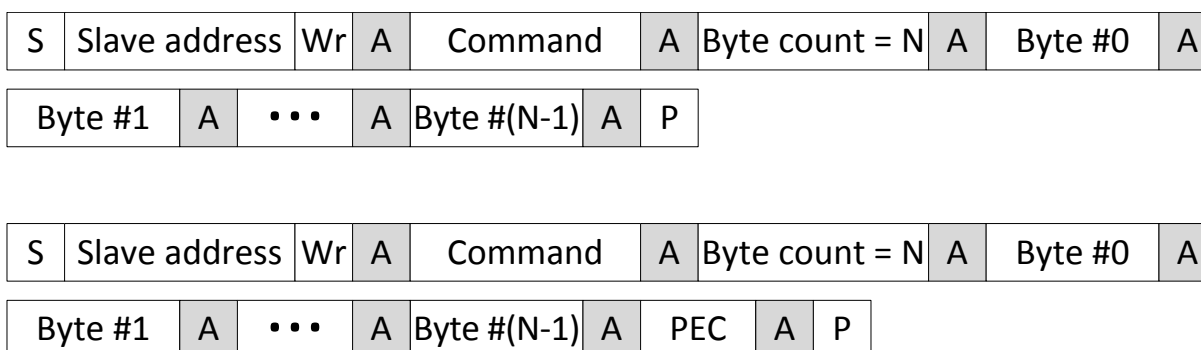
After programming the PMBMC register, the PMBus module transmits the Process Call Message. The firmware can wait for an End of Message interrupt from the module to determine the validity of the message. Upon the receipt of the EOM, the PMBSTS register should indicate the receipt of 2 bytes from the Read Word portion of the Process Call message and the status of the Slave acknowledgment of the transmit data. If PEC processing is enabled, the PEC\_VAL bit within the PMBSTS register indicates the accuracy of the PEC byte received from the slave during the Read Word part of the message.

The PRC\_CALL bit within the PMBMC register must be disabled for the next non-Process Call message. Please note that any write to the PMBMC register initiates a message, so reconfiguration of the master is not recommended until the firmware requires a new message to be transmitted.

#### 25.4.2.7 Block Write

The Block Write protocol is similar to a Write Word in its structure, with the exception of transmission of more than 2 data bytes in the message. Additionally, the first data byte following the command byte specifies the length of the block of data bytes. As with a majority of the message protocols, the PEC byte can be appended to the end of the write data to the slave.

To initiate a Block Write message on the bus, the PMBMC register is programmed with the block length in the Byte Count bits. The block length is the number of data bytes, excluding the command byte and the first data byte that contains the block length. The PMBus module will automatically insert the block length into the message if the number of data bytes specified by the firmware exceeds 2. The initial write data is loaded into the PMBTXBUF register. With bits 7-0 representing the command byte, the remaining 3 bytes represent the first three data bytes following the block length.

**Figure 25-21. Block Write Message with and without PEC**




Following programming of the PMBMC register, the Block Write message is transmitted. If the block length exceeds three bytes, the PMBus module will provide a data request interrupt, indicating the need for additional data bytes in the PMBTXBUF register. The PMBus module assumes that if more than 4 bytes are needed to complete the message, the firmware will utilize all 4 bytes when programming the PMBTXBUF register. If less than 4 bytes are needed to finish the Block Write message, the firmware only needs to program the appropriate bits of the PMBTXBUF register.

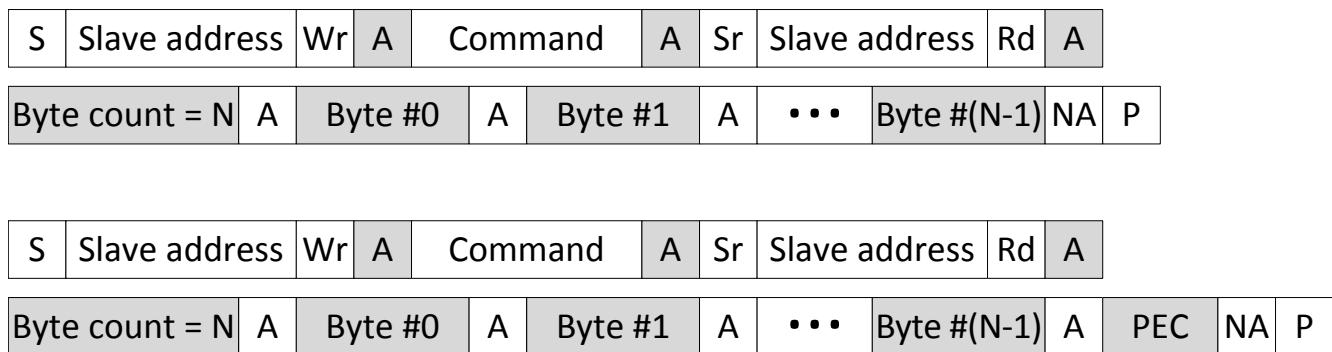
Upon completion of the message, the PMBus module issues an EOM interrupt. The PMBSTS register can be checked to verify the slave accepted the block of write data.

### 25.4.2.8 Block Read

The Block Read protocol is similar to a Read Word in its structure, with the exception that there are more than 2 data bytes received from the slave. The first data byte transmitted by the slave represents the block length of the data being written by the slave. If PEC processing is enabled, the slave appends a PEC byte to the end of the message.

To initiate a Block Read message on the PMBus, the PMBMC register is programmed with the block length in the Byte Count bits. This count excludes the command byte, any slave address and the block length bytes in the message. The command byte to be transmitted to the slave is written into bits 7-0 of the PMBTXBUF register prior to the programming of the PMBMC register.

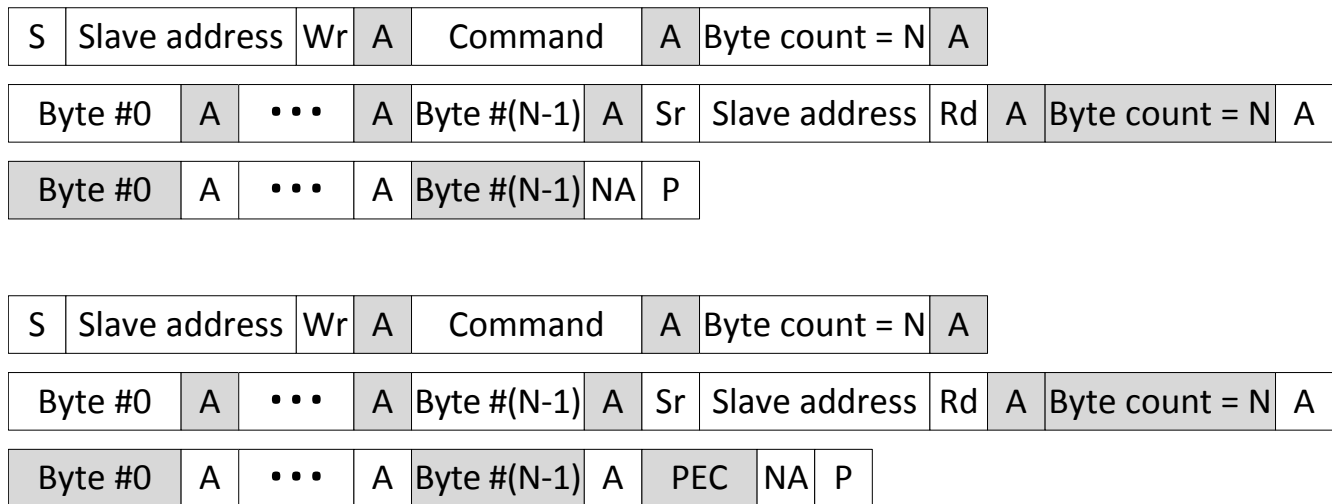
**Figure 25-22. Block Read Message with and without PEC**



After configuring the PMBMC register, the Block Read message is transmitted. The module interrupts the firmware upon receipt of 4 data bytes from the slave. If the block length is 3, the EOM interrupt will be received concurrently with the data ready interrupt. Otherwise, only a data ready interrupt is asserted, indicating 4 bytes are ready for reading by the firmware. At the end of the message, less than 4 bytes may be stored in the PMBRXBUF register. The RX Byte Count bits in the PMBSTS register indicate the number of bytes available in the final data transfer. The firmware may verify the received PEC upon detection of the End of Message interrupt.

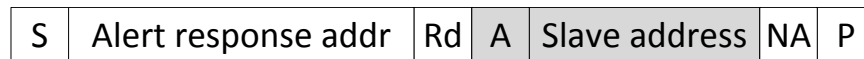
### 25.4.2.9 Block Write-Block Read Process Call

The Block Read-Block Write Process Call protocol combines the Block Write and Block Read protocols, removing the stop condition between the two messages. The operation of the master is similar to a Block Write operation. Loading the block length into the byte count bits of the PMBMC register provides the length of the Block Write portion of the message. In addition, the PRC\_CALL bit within the PMBMC register must be enabled. Upon completion of the Block Write part of the message, the PMBus module will automatically issue a Repeated Start condition on the PMBus and start transmission of the Block Read portion of the message. Operation of the PMBus module after the Repeated Start condition is the same as it would be in a simple Block Read Message.

**Figure 25-23. Block Write-Block Read Process Call Message with and without PEC**


#### 25.4.2.10 Alert Response

The Alert Response Message is utilized when the master detects an alert condition from a slave. In master mode, the Alert Response Message is simply a Receive Byte message with PEC disable and the slave address set to 0xC (Alert Response address). The PMBus module detects the alert condition on an input and interrupts the firmware indicating the assertion of an alert condition (slave desires to communicate with master). Programming the PMBMC register with the Alert Response address initiates the Alert Response message and provides the device address of the slave requesting service. The device address will be found in the PMBRXBUF register following receipt of the EOM interrupt.

**Figure 25-24. Alert Response Message**


#### 25.4.2.11 Extended Command

The PMBus module provides support for extended commands which allow for an extra 256 command codes. By asserting the EXT\_CMD bit within the PMBMC register, two command bytes are transmitted on the message protocol. Extended commands can be added to the Read Byte, Read Word, Write Byte and Write Word protocols. Operation of the PMBus module in extended command mode is similar to these formats. In programming the write data or first part of the read message, the second command byte is loaded into bits 15-8 of the PMBTXBUF register with the remaining data bytes. The remaining operation of the module is identical to the previous protocols, except for the inclusion of a Repeated Start condition and slave address in the write messages. No support is required by firmware for these additional bytes in the write messages. The module will interpret the EXT\_CMD bit and make the appropriate format changes.

**Figure 25-25. Extended Command Write Message with and without PEC**

S	Slave address	Wr	A	Extended command	A	Command	A
---	---------------	----	---	------------------	---	---------	---

Sr	Slave address	Wr	A	Byte #0	A	P
----	---------------	----	---	---------	---	---

S	Slave address	Wr	A	Extended command	A	Command	A
---	---------------	----	---	------------------	---	---------	---

Sr	Slave address	Wr	A	Byte #0	A	PEC	A	P
----	---------------	----	---	---------	---	-----	---	---

S	Slave address	Wr	A	Extended command	A	Command	A
---	---------------	----	---	------------------	---	---------	---

Sr	Slave address	Wr	A	Byte #0	A	Byte #1	A	P
----	---------------	----	---	---------	---	---------	---	---

S	Slave address	Wr	A	Extended command	A	Command	A
---	---------------	----	---	------------------	---	---------	---

Sr	Slave address	Wr	A	Byte #0	A	Byte #1	A	PEC	A	P
----	---------------	----	---	---------	---	---------	---	-----	---	---

**Figure 25-26. Extended Command Read Message with and without PEC**

S	Slave address	Wr	A	Extended command	A	Command	A
---	---------------	----	---	------------------	---	---------	---

Sr	Slave address	Rd	A	Byte #0	NA	P
----	---------------	----	---	---------	----	---

S	Slave address	Wr	A	Extended command	A	Command	A
---	---------------	----	---	------------------	---	---------	---

Sr	Slave address	Rd	A	Byte #0	A	PEC	NA	P
----	---------------	----	---	---------	---	-----	----	---

S	Slave address	Wr	A	Extended command	A	Command	A
---	---------------	----	---	------------------	---	---------	---

Sr	Slave address	Rd	A	Byte #0	A	Byte #1	NA	P
----	---------------	----	---	---------	---	---------	----	---

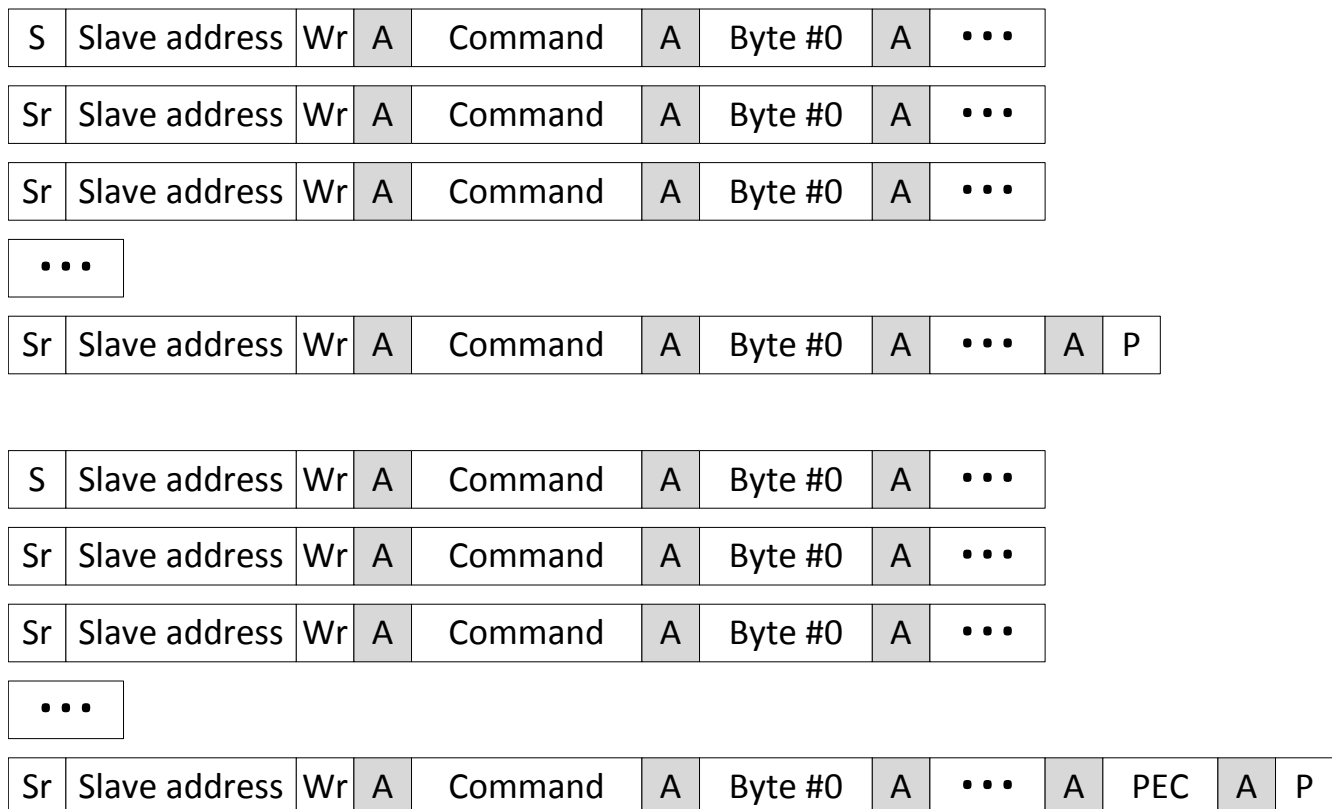
S	Slave address	Wr	A	Extended command	A	Command	A
---	---------------	----	---	------------------	---	---------	---

Sr	Slave address	Rd	A	Byte #0	A	Byte #1	A	PEC	NA	P
----	---------------	----	---	---------	---	---------	---	-----	----	---

### 25.4.2.12 Group Command

The Group Command Protocol is used to send commands to more than one device within the same message. When devices on the bus detect the stop condition at the conclusion of the Group Command message, the received commands are executed concurrently. To initiate a Group Command, the GRP\_CMD bit within the PMBMC register must be set when programming the slave address for the first device in the message. The rest of the message is processed as a write byte/word message. At the conclusion of the first part of the Group Command message, the firmware programs the next device address in the PMBMC register. The PMBus module will send a repeated start on the bus and begin the next part of the message. When programming the last device address of the Group Command message, the firmware must disable the GRP\_CMD bit when programming the PMBMC register.

**Figure 25-27. Group Command Message with and without PEC**



## 25.5 PMBus Registers

This section describes the Power-Management Bus module Registers.

### 25.5.1 *PMBus Base Addresses*

**Table 25-1. PMBus Base Address Table**

Device Registers	Register Name	Start Address	End Address
PmbusaRegs	PMBUS_REGS	0x0000_6400	0x0000_641F

## 25.5.2 PMBUS\_REGS Registers

Table 25-2 lists the PMBUS\_REGS registers. All register offset addresses not listed in Table 25-2 should be considered as reserved locations and the register contents should not be modified.

**Table 25-2. PMBUS\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	PMBMC	PMBUS Master Mode Control Register	EALLOW	<a href="#">Go</a>
2h	PMBTXBUF	PMBUS Transmit Buffer		<a href="#">Go</a>
4h	PMBRXBUF	PMBUS Receive buffer		<a href="#">Go</a>
6h	PMBACK	PMBUS Acknowledge Register		<a href="#">Go</a>
8h	PMBSTS	PMBUS Status Register		<a href="#">Go</a>
Ah	PMBINTM	PMBUS Interrupt Mask Register	EALLOW	<a href="#">Go</a>
Ch	PMBSC	PMBUS Slave Mode Configuration Register	EALLOW	<a href="#">Go</a>
Eh	PMBHSA	PMBUS Hold Slave Address Register		<a href="#">Go</a>
10h	PMBCTRL	PMBUS Control Register	EALLOW	<a href="#">Go</a>
12h	PMBTIMCTL	PMBUS Timing Control Register	EALLOW	<a href="#">Go</a>
14h	PMBTIMCLK	PMBUS Clock Timing Register	EALLOW	<a href="#">Go</a>
16h	PMBTIMSTSETUP	PMBUS Start Setup Time Register	EALLOW	<a href="#">Go</a>
18h	PMBTIMBIDLE	PMBUS Bus Idle Time Register	EALLOW	<a href="#">Go</a>
1Ah	PMBTIMLOWTIMEOUT	PMBUS Clock Low Timeout Value Register	EALLOW	<a href="#">Go</a>
1Ch	PMBTIMHIGHTIMEOUT	PMBUS Clock High Timeout Value Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 25-3 shows the codes that are used for access types in this section.

**Table 25-3. PMBUS\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
RC	R C	Read to Clear
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**25.5.2.1 PMBMC Register (Offset = 0h) [reset = 0h]**

 PMBMC is shown in [Figure 25-28](#) and described in [Table 25-4](#).

 Return to the [Summary Table](#).

PMBUS Master Mode Control Register

**Figure 25-28. PMBMC Register**

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

**Table 25-4. PMBMC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-21	RESERVED	R	0h	Reserved
20	PRC_CALL	R/W	0h	0 = Default state for all messages besides Process Call message 1 = Enables transmission of Process Call message Reset type: SYSRSn
19	GRP_CMD	R/W	0h	0 = Default state for all messages besides Group Command message 1 = Enables transmission of Group Command message Reset type: SYSRSn
18	PEC_ENA	R/W	0h	0 = Disables PEC processing 1 = Enables PEC byte transmission/reception Reset type: SYSRSn
17	EXT_CMD	R/W	0h	0 = Use 1 byte for Command Code 1 = Use 2 bytes for Command Code Reset type: SYSRSn
16	CMD_ENA	R/W	0h	0 = Disables use of command code on Master initiated messages ( 1 = Enables use of command code on Master initiated messages Reset type: SYSRSn
15-8	BYTE_COUNT	R/W	0h	Indicates number of data bytes transmitted in current message. Byte count does not include any device addresses, command words or block lengths in block messages. In block messages, the PMBus Interface automatically inserts the block length into the message based on the byte count setting. The firmware only needs to load the address, command words and data to be transmitted. PMBus Interface supports byte writes up to 255 bytes. Reset type: SYSRSn
7-1	SLAVE_ADDR	R/W	0h	Specifies the address of the slave to which the current message is directed towards. Reset type: SYSRSn

**Table 25-4. PMBMC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	RW	R/W	0h	0 = Message is a write transaction (data from Master to Slave) 1 = Message is a read transaction (data from Slave to Master) Reset type: SYSRSn



### 25.5.2.2 PMBTXBUF Register (Offset = 2h) [reset = 0h]

PMBTXBUF is shown in [Figure 25-29](#) and described in [Table 25-5](#).

Return to the [Summary Table](#).

PMBUS Transmit Buffer

**Figure 25-29. PMBTXBUF 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
																	TXDATA														
R/W-0h																															

**Table 25-5. PMBTXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	TXDATA	R/W	0h	Bits 31-24: BYTE3 - Last data byte transmitted from Transmit Data Buffer Bits 23-16: BYTE2 - Third data byte transmitted from Transmit Data Buffer Bits 15-8: BYTE1 - Second data byte transmitted from Transmit Data Buffer Bits 7-0: BYTE0 - First data byte transmitted from Transmit Data Buffer Reset type: SYSRSn

### 25.5.2.3 PMBRXBUF Register (Offset = 4h) [reset = 0h]

PMBRXBUF is shown in [Figure 25-30](#) and described in [Table 25-6](#).

Return to the [Summary Table](#).

PMBUS Receive buffer

**Figure 25-30. PMBRXBUF 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
RXDATA																															
R-0h																															

**Table 25-6. PMBRXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RXDATA	R	0h	Bits 31-24: BYTE3 - Last data byte received in Receive Data Buffer Bits 23-16: BYTE2 - Third data byte received in Receive Data Buffer Bits 15-8: BYTE1 - Second data byte received in Receive Data Buffer Bits 7-0: BYTE0 - First data byte received in Receive Data Buffer Reset type: SYSRSn

**25.5.2.4 PMBACK Register (Offset = 6h) [reset = 0h]**

PMBACK is shown in [Figure 25-31](#) and described in [Table 25-7](#).

Return to the [Summary Table](#).

PMBUS Acknowledge Register

**Figure 25-31. PMBACK 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															ACK
R-0h															R/W-0h

**Table 25-7. PMBACK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ACK	R/W	0h	0 = NACK received data 1 = Acknowledge received data, bit clears upon issue of ACK on PMBus Reset type: SYSRSn

**25.5.2.5 PMBSTS Register (Offset = 8h) [reset = 00340000h]**

 PMBSTS is shown in [Figure 25-32](#) and described in [Table 25-8](#).

 Return to the [Summary Table](#).

PMBUS Status Register

**Figure 25-32. PMBSTS Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED		SCL_RAW	SDA_RAW	CONTROL_RAW	ALERT_RAW	CONTROL_EDGE	ALERT_EDGE
R-0h		R-1h	R-1h	R-0h	R-1h	RC-0h	RC-0h
15	14	13	12	11	10	9	8
MASTER	LOST_ARB	BUS_FREE	UNIT_BUSY	RPT_START	SLAVE_ADDR_READY	CLK_HIGH_DETECTED	CLK_LOW_TIMEOUT
RC-0h	RC-0h	RC-0h	RC-0h	RC-0h	RC-0h	RC-0h	RC-0h
7	6	5	4	3	2	1	0
PEC_VALID	NACK	EOM	DATA_REQUEST	DATA_READY	RD_BYTE_COUNT		
RC-0h	RC-0h	RC-0h	RC-0h	RC-0h	RC-0h		

**Table 25-8. PMBSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0h	Reserved
21	SCL_RAW	R	1h	0 = PMBus clock pin observed at logic level low 1 = PMBus clock pin observed at logic level high Reset type: SYSRSn
20	SDA_RAW	R	1h	0 = PMBus data pin observed at logic level low 1 = PMBus data pin observed at logic level high Reset type: SYSRSn
19	CONTROL_RAW	R	0h	0 = Control pin observed at logic level low 1 = Control pin observed at logic level high Reset type: SYSRSn
18	ALERT_RAW	R	1h	0 = Alert pin observed at logic level low 1 = Alert pin observed at logic level high Reset type: SYSRSn
17	CONTROL_EDGE	RC	0h	0 = Control pin has not transitioned 1 = Control pin has been asserted by another device on PMBus Reset type: SYSRSn
16	ALERT_EDGE	RC	0h	0 = Alert pin has not transitioned 1 = Alert pin has been asserted by another device on PMBus Reset type: SYSRSn
15	MASTER	RC	0h	0 = PMBus Interface in Slave Mode or Idle Mode 1 = PMBus Interface in Master Mode Reset type: SYSRSn
14	LOST_ARB	RC	0h	0 = Master has attained control of PMBus 1 = Master has lost arbitration and control of PMBus Reset type: SYSRSn

**Table 25-8. PMBSTS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	BUS_FREE	RC	0h	0 = PMBus processing current message 1 = PMBus available for new message Reset type: SYSRSn
12	UNIT_BUSY	RC	0h	0 = PMBus Interface is idle, ready to transmit/receive message 1 = PMBus Interface is busy, processing current message Reset type: SYSRSn
11	RPT_START	RC	0h	0 = No Repeated Start received by interface 1 = Repeated Start condition received by interface Reset type: SYSRSn
10	SLAVE_ADDR_READY	RC	0h	0 = Indicates no slave address is available for reading 1 = Slave address ready to be read from Receive Data Register (Bits 6:0) Reset type: SYSRSn
9	CLK_HIGH_DETECTED	RC	0h	0 = No Clock High condition detected 1 = Clock High exceeded 50us during message Reset type: SYSRSn
8	CLK_LOW_TIMEOUT	RC	0h	0 = No clock low timeout detected 1 = Clock low timeout detected, clock held low for greater than 35ms Reset type: SYSRSn
7	PEC_VALID	RC	0h	0 = Received PEC not valid (if EOM is asserted) 1 = Received PEC is valid Note: PEC_VALID status is don't care during the message. This will have a valid value only after EOM. Reset type: SYSRSn
6	NACK	RC	0h	0 = Data transmitted has been accepted by receiver 1 = Receiver has not accepted transmitted data Reset type: SYSRSn
5	EOM	RC	0h	0 = Message still in progress or PMBus in idle state. 1 = End of current message detected Reset type: SYSRSn
4	DATA_REQUEST	RC	0h	0 = No data needed by PMBus Interface 1 = PMBus Interface request additional data. PMBus clock stretching enabled to stall bus Reset type: SYSRSn
3	DATA_READY	RC	0h	0 = No data available for reading by processor 1 = PMBus Interface read buffer full, firmware required to read data prior to further bus activity. PMBus clock stretching enabled to stall bus until data is read by firmware. Reset type: SYSRSn
2-0	RD_BYTE_COUNT	RC	0h	0 = No received data 1 = 1 byte received. Data located in Receive Data Register, Bits 7-0 2 = 2 bytes received. Data located in Receive Data Register, Bits 15-0 3 = 3 bytes received. Data located in Receive Data Register, Bits 23-0 4 = 4 bytes received. Data located in Receive Data Register, Bits 31-0 Reset type: SYSRSn

**25.5.2.6 PMBINTM Register (Offset = Ah) [reset = 3FFh]**

PMBINTM is shown in [Figure 25-33](#) and described in [Table 25-9](#).

Return to the [Summary Table](#).

PMBUS Interrupt Mask Register

**Figure 25-33. PMBINTM 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						CLK_HIGH_DETECT	LOST_ARB
R-0h						R/W-1h	R/W-1h
7	6	5	4	3	2	1	0
CONTROL	ALERT	EOM	SLAVE_ADDR_READY	DATA_REQUEST	DATA_READY	BUS_LOW_TIME_OUT	BUS_FREE
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h

**Table 25-9. PMBINTM Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	CLK_HIGH_DETECT	R/W	1h	0 = Generates interrupt if clock high exceeds 50us during message 1 = Disables interrupt generation for Clock High detection Reset type: SYSRSn
8	LOST_ARB	R/W	1h	0 = Generates interrupt upon assertion of Lost Arbitration flag 1 = Disables interrupt generation upon assertion of Lost Arbitration flag Reset type: SYSRSn
7	CONTROL	R/W	1h	0 = Generates interrupt upon assertion of Control flag 1 = Disables interrupt generation upon assertion of Control flag Reset type: SYSRSn
6	ALERT	R/W	1h	0 = Generates interrupt upon assertion of Alert flag 1 = Disables interrupt generation upon assertion of Alert flag Reset type: SYSRSn
5	EOM	R/W	1h	0 = Generates interrupt upon assertion of End of Message flag 1 = Disables interrupt generation upon assertion of End of Message flag Reset type: SYSRSn
4	SLAVE_ADDR_READY	R/W	1h	0 = Generates interrupt upon assertion of Slave Address Ready flag 1 = Disables interrupt generation upon assertion of Slave Address Ready flag Reset type: SYSRSn
3	DATA_REQUEST	R/W	1h	0 = Generates interrupt upon assertion of Data Request flag 1 = Disables interrupt generation upon assertion of Data Request flag Reset type: SYSRSn

**Table 25-9. PMBINTM Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	DATA_READY	R/W	1h	0 = Generates interrupt upon assertion of Data Ready flag 1 = Disables interrupt generation upon assertion of Data Ready flag Reset type: SYSRSn
1	BUS_LOW_TIMEOUT	R/W	1h	0 = Generates interrupt upon assertion of Clock Low Timeout flag 1 = Disables interrupt generation upon assertion of Clock Low Timeout flag Reset type: SYSRSn
0	BUS_FREE	R/W	1h	0 = Generates interrupt upon assertion of Bus Free flag 1 = Disables interrupt generation upon assertion of Bus Free flag Reset type: SYSRSn

**25.5.2.7 PMBSC Register (Offset = Ch) [reset = 00607F7Ch]**

PMBSC is shown in [Figure 25-34](#) and described in [Table 25-10](#).

Return to the [Summary Table](#).

PMBUS Slave Mode Configuration Register

**Figure 25-34. PMBSC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED	RX_BYTE_ACK_CNT		MAN_CMD	TX_PEC	TX_COUNT		
R-0h	R/W-3h		R/W-0h	R/W-0h	R/W-0h		
15	14	13	12	11	10	9	8
PEC_ENA	SLAVE_MASK						
R/W-0h	R/W-7Fh						
7	6	5	4	3	2	1	0
MAN_SLAVE_ACK	SLAVE_ADDR						
R/W-0h	R/W-7Ch						

**Table 25-10. PMBSC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0h	Reserved
22-21	RX_BYTE_ACK_CNT	R/W	3h	Configures number of data bytes to automatically acknowledge when receiving data in slave mode. 00 = 1 byte received by slave. Firmware is required to manually acknowledge every received byte. 01 = 2 bytes received by slave. Hardware automatically acknowledges the first received byte. Firmware is required to manually acknowledge after the second received byte. 10 = 3 bytes received by slave. Hardware automatically acknowledges the first 2 received bytes. Firmware is required to manually acknowledge after the third received byte. 11 = 4 bytes received by slave. Hardware automatically acknowledges the first 3 received bytes. Firmware is required to manually acknowledge after the fourth received byte Reset type: SYSRSn
20	MAN_CMD	R/W	0h	0 = Slave automatically acknowledges received command code 1 = Data Request flag generated after receipt of command code, firmware required to issue ACK to continue message Reset type: SYSRSn
19	TX_PEC	R/W	0h	Asserted when the slave needs to send a PEC byte at end of message. PMBus Interface will transmit the calculated PEC byte after transmitting the number of data bytes indicated by TX Byte Cnt(Bits 18:16). Reset type: SYSRSn



**Table 25-10. PMBSC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-16	TX_COUNT	R/W	0h	<p>0 = No bytes valid</p> <p>1 = One byte valid, Byte #0 (Bits 7:0 of Transmit Data Register)</p> <p>2 = Two bytes valid, Bytes #0 and #1 (Bits 15:0 of Transmit Data Register)</p> <p>3 = Three bytes valid, Bytes #0-2 (Bits 23:0 of Transmit Data Register)</p> <p>4 = Four bytes valid, Bytes #0-3 (Bits 31:0 of Transmit Data Register)</p> <p>Reset type: SYSRSn</p>
15	PEC_ENA	R/W	0h	<p>0 = PEC processing disabled</p> <p>1 = PEC processing enabled</p> <p>Reset type: SYSRSn</p>
14-8	SLAVE_MASK	R/W	7Fh	<p>Used in address detection, the slave mask enables acknowledgement of multiple device addresses by the slave. Writing a '0' to a bit within the slave mask enables the corresponding bit in the slave address to be either '1' or '0' and still allow for a match. Writing a '0' to all bits in the mask enables the PMBus Interface to acknowledge any device address. Upon power-up, the slave mask defaults to 7Fh, indicating the slave will only acknowledge the address programmed into the Slave Address (Bits 6-0).</p> <p>Reset type: SYSRSn</p>
7	MAN_SLAVE_ACK	R/W	0h	<p>0 = Slave automatically acknowledges device address specified in SLAVE_ADDR, Bits 6:0</p> <p>1 = Enables the Manual Slave Address Acknowledgement Mode. Firmware is required to read received address and acknowledge on every message</p> <p>Note: When bit 31 (I2C_mode) of PMBCTRL register is set it is recommended to use manual acknowledging of slave address only (MAN_SLAVE_ACK =1).</p> <p>Reset type: SYSRSn</p>
6-0	SLAVE_ADDR	R/W	7Ch	<p>Configures the current device address of the slave. Used in automatic slave address acknowledge mode (default mode). The PMBus Interface will compare the received device address with the value stored in the Slave Address bits and the mask configured in the Slave Mask bits. If matching, the slave will acknowledge the device address.</p> <p>Reset type: SYSRSn</p>

### 25.5.2.8 PMBHSA Register (Offset = Eh) [reset = 0h]

PMBHSA is shown in [Figure 25-35](#) and described in [Table 25-11](#).

Return to the [Summary Table](#).

PMBUS Hold Slave Address Register

**Figure 25-35. PMBHSA 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
SLAVE_ADDR							SLAVE_RW
R-0h							R-0h

**Table 25-11. PMBHSA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-1	SLAVE_ADDR	R	0h	Stored device address acknowledged by the slave Reset type: SYSRSn
0	SLAVE_RW	R	0h	Stored R/W bit from address acknowledged by the slave 0 = Write Access 1 = Read Access Reset type: SYSRSn

### 25.5.2.9 PMBCTRL Register (Offset = 10h) [reset = 00200000h]

PMBCTRL is shown in [Figure 25-36](#) and described in [Table 25-12](#).

Return to the [Summary Table](#).

PMBUS Control Register

**Figure 25-36. PMBCTRL Register**

31		30		29		28		27		26		25		24	
I2CMODE		RESERVED						CLKDIV							
R/W-0h		R-0h						R/W-0h							
23		22		21		20		19		18		17		16	
CLKDIV		MASTER_EN		SLAVE_EN		CLK_LO_DIS		IBIAS_B_EN		IBIAS_A_EN		SCL_DIR		SCL_VALUE	
R/W-0h		R/W-0h		R/W-1h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15		14		13		12		11		10		9		8	
SCL_MODE		SDA_DIR		SDA_VALUE		SDA_MODE		CNTL_DIR		CNTL_VALUE		CNTL_MODE		ALERT_DIR	
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	
ALERT_VALUE		ALERT_MODE		CNTL_INT_EDGE		RESERVED		FAST_MODE		BUS_LO_INT_EDGE		ALERT_EN		RESET	
R/W-0h		R/W-0h		R/W-0h		R-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 25-12. PMBCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	I2CMODE	R/W	0h	0 = PMBUS mode 1 = I2C mode Reset type: SYSRSn
30-28	RESERVED	R	0h	Reserved
27-23	CLKDIV	R/W	0h	The clock to the PMBUS transmit/receive FSMs (FSM_CLK) is divided version of the SYSCLK clock. Frequency(FSM_CLK) = Frequency(SYSCLK)/(CLKDIV+1) Note: FSM_CLK should be less than (or) equal to 10MHz. Reset type: SYSRSn
22	MASTER_EN	R/W	0h	0 = Disables PMBus Master capability 1 = Enables PMBus Master capability Reset type: SYSRSn
21	SLAVE_EN	R/W	1h	0 = Disables PMBus Slave capability 1 = Enables PMBus Slave capability Reset type: SYSRSn
20	CLK_LO_DIS	R/W	0h	0 = Clock Low Timeout Enabled 1 = Clock Low Timeout Disabled Reset type: SYSRSn
19	IBIAS_B_EN	R/W	0h	0 = Disables Current Source for PMBUS address detection thru ADC 1 = Enables Current Source for PMBUS address detection thru ADC Reset type: SYSRSn
18	IBIAS_A_EN	R/W	0h	0 = Disables Current Source for PMBUS address detection thru ADC 1 = Enables Current Source for PMBUS address detection thru ADC Reset type: SYSRSn
17	SCL_DIR	R/W	0h	0 = PMBus clock pin configured as output 1 = PMBus clock pin configured as input Reset type: SYSRSn

**Table 25-12. PMBCTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	SCL_VALUE	R/W	0h	0 = PMBus clock pin driven low in GPIO Mode 1 = PMBus clock pin driven high in GPIO Mode Reset type: SYSRSn
15	SCL_MODE	R/W	0h	0 = PMBus clock pin configured in functional mode 1 = PMBus clock pin configured as GPIO Reset type: SYSRSn
14	SDA_DIR	R/W	0h	0 = PMBus data pin configured as output 1 = PMBus data pin configured as input Reset type: SYSRSn
13	SDA_VALUE	R/W	0h	0 = PMBus data pin driven low in GPIO Mode 1 = PMBus data pin driven high in GPIO Mode Reset type: SYSRSn
12	SDA_MODE	R/W	0h	0 = PMBus data pin driven low in GPIO Mode 1 = PMBus data pin driven high in GPIO Mode Reset type: SYSRSn
11	CNTL_DIR	R/W	0h	0 = Control pin configured as output 1 = Control pin configured as input Reset type: SYSRSn
10	CNTL_VALUE	R/W	0h	0 = Control pin driven low in GPIO Mode 1 = Control pin driven high in GPIO Mode Reset type: SYSRSn
9	CNTL_MODE	R/W	0h	0 = Control pin configured in functional mode (Default) 1 = Control pin configured as GPIO Reset type: SYSRSn
8	ALERT_DIR	R/W	0h	0 = Alert pin configured as output 1 = Alert pin configured as input Reset type: SYSRSn
7	ALERT_VALUE	R/W	0h	0 = Alert pin driven low in GPIO Mode 1 = Alert pin driven high in GPIO Mode Reset type: SYSRSn
6	ALERT_MODE	R/W	0h	0 = Alert pin configured in functional mode 1 = Aler3 pin configured as GPIO Reset type: SYSRSn
5	CNTL_INT_EDGE	R/W	0h	0 = Interrupt generated on falling edge of Control 1 = Interrupt generated on rising edge of Control Reset type: SYSRSn
4	RESERVED	R	0h	Reserved
3	FAST_MODE	R/W	0h	0 = Standard 100 KHz mode enabled 1 = Fast Mode enabled (400KHz operation on PMBus) Reset type: SYSRSn
2	BUS_LO_INT_EDGE	R/W	0h	0 = Interrupt generated on rising edge of clock low timeout 1 = Interrupt generated on falling edge of clock low timeout Reset type: SYSRSn
1	ALERT_EN	R/W	0h	0 = PMBus Alert is not driven by slave, pulled up high on PMBus 1 = PMBus Alert driven low by slave Reset type: SYSRSn

**Table 25-12. PMBCTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	RESET	R/W	0h	0 = No reset of internal state machines (Default) 1 = Control state machines are reset to initial states Note: Status register PMBSTS should be explicitly cleared by reading the register after softreset as this will not be cleared by Software Reset. Reset type: SYSRSn

**25.5.2.10 PMBTIMCTL Register (Offset = 12h) [reset = 0h]**

PMBTIMCTL is shown in [Figure 25-37](#) and described in [Table 25-13](#).

Return to the [Summary Table](#).

PMBUS Timing Control Register

**Figure 25-37. PMBTIMCTL 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							TIM_OVERRIDE
R-0h							R/W-0h

**Table 25-13. PMBTIMCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	TIM_OVERRIDE	R/W	0h	0 PMBUS FSMs uses the default settings of the timing parameters. 1 PMBUS FSMs would use the settings in following registers: * PMBTIMCLK * PMBTIMSTSETUP * PMBTIMBIDLE * PMBTIMLOWTIMOUT * PMBTIMHIGHTIMOUT Reset type: SYSRSn

**25.5.2.11 PMBTIMCLK Register (Offset = 14h) [reset = 0060002Fh]**

PMBTIMCLK is shown in [Figure 25-38](#) and described in [Table 25-14](#).

Return to the [Summary Table](#).

PMBUS Clock Timing Register

**Figure 25-38. PMBTIMCLK Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								CLK_FREQ							
R-0h								R/W-60h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CLK_HIGH_LIMIT							
R-0h								R/W-2Fh							

**Table 25-14. PMBTIMCLK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	CLK_FREQ	R/W	60h	Defines the number of PMBUS FSM input clock in the PMBUS master clock period. Number of FSM clocks in the one clock period = (CLK_HIGH_LIMIT+4) Reset type: SYSRSn
15-8	RESERVED	R	0h	Reserved
7-0	CLK_HIGH_LIMIT	R/W	2Fh	Defines the number of PMBUS FSM input clock in the PMBUS master clock high pulse. Number of FSM clocks in the one clock high pulse = (CLK_HIGH_LIMIT+3) Reset type: SYSRSn

**25.5.2.12 PMBTIMSTSETUP Register (Offset = 16h) [reset = 2Fh]**

PMBTIMSTSETUP is shown in [Figure 25-39](#) and described in [Table 25-15](#).

Return to the [Summary Table](#).

PMBUS Start Setup Time Register

**Figure 25-39. PMBTIMSTSETUP 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																								TSU_STA							
R-0h																								R/W-2Fh							

**Table 25-15. PMBTIMSTSETUP Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	TSU_STA	R/W	2Fh	Determines the Setup time between last rise edge of the PMBUS master clock and the next start edge, TSU_STA value defines the setup time in terms of PMBUS FSM clock cycles. Reset type: SYSRSn



**25.5.2.13 PMBTIMBIDLE Register (Offset = 18h) [reset = 1F3h]**

PMBTIMBIDLE is shown in [Figure 25-40](#) and described in [Table 25-16](#).

Return to the [Summary Table](#).

PMBUS Bus Idle Time Register

**Figure 25-40. PMBTIMBIDLE 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														BUSIDLE																	
R-0h														R/W-1F3h																	

**Table 25-16. PMBTIMBIDLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9-0	BUSIDLE	R/W	1F3h	Determines the duration for which PMBUS clock and Data are 1 , to conclude that the bus is IDLE. BUSIDLE value is in terms of number of PMBUS FSM clock cycles. Reset type: SYSRSn

**25.5.2.14 PMBTIMLOWTIMOUT Register (Offset = 1Ah) [reset = 0005572Fh]**

PMBTIMLOWTIMOUT is shown in [Figure 25-41](#) and described in [Table 25-17](#).

Return to the [Summary Table](#).

PMBUS Clock Low Timeout Value Register

**Figure 25-41. PMBTIMLOWTIMOUT 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												CLKLOWTIMOUT																			
R-0h												R/W-0005572Fh																			

**Table 25-17. PMBTIMLOWTIMOUT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-0	CLKLOWTIMOUT	R/W	0005572Fh	Determines the duration for which PMBUS clock if low , will result in a clock low timeout condition. CLKLOWTIMOUT value is in terms of number of PMBUS FSM clock cycles. Reset type: SYSRSn

**25.5.2.15 PMBTIMHIGHTIMOUT Register (Offset = 1Ch) [reset = 1F3h]**

PMBTIMHIGHTIMOUT is shown in [Figure 25-42](#) and described in [Table 25-18](#).

Return to the [Summary Table](#).

PMBUS Clock High Timeout Value Register

**Figure 25-42. PMBTIMHIGHTIMOUT 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							CLKHIGHTIMOUT								
R-0h							R/W-1F3h								

**Table 25-18. PMBTIMHIGHTIMOUT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9-0	CLKHIGHTIMOUT	R/W	1F3h	Determines the duration for which PMBUS clock if high , will result in a clock high timeout condition. CLKHIGHTIMOUT value is in terms of number of PMBUS FSM clock cycles. Reset type: SYSRSn

## Controller Area Network (CAN)

---



---

This chapter describes the Controller Area Network (CAN) module. CAN is a serial communications protocol which efficiently supports distributed real-time control with a high level of reliability. The CAN module supports bit rates up to 1 Mbit/s and is compliant with the ISO11898-1 (CAN 2.0B) protocol specification.

Further information about this module can be found in the following document(s):

- [Calculator for CAN Bit Timing Parameters](#)

Topic	Page
26.1 Introduction .....	2373
26.2 Configuring Device Pins .....	2375
26.3 Address/Data Bus Bridge.....	2375
26.4 Operating Modes .....	2378
26.5 Multiple Clock Source .....	2382
26.6 Interrupt Functionality .....	2383
26.7 DMA Functionality .....	2385
26.8 Parity Check Mechanism .....	2385
26.9 Debug Mode .....	2386
26.10 Module Initialization .....	2386
26.11 Configuration of Message Objects .....	2387
26.12 Message Handling .....	2388
26.13 CAN Bit Timing .....	2393
26.14 Message Interface Register Sets .....	2401
26.15 Message RAM .....	2404
26.16 CAN Registers.....	2408

## 26.1 Introduction

This device uses the CAN IP known as DCAN.

### 26.1.1 Features

The CAN module implements the following features:

- Complies with ISO11898-1 (Bosch® CAN protocol specification 2.0 A and B)
- Bit rates up to 1 Mbps
- Multiple clock sources
- 32 message objects (“message objects” are also referred to as “mailboxes” in this document; the two terms are used interchangeably), each with the following properties:
  - Configurable as receive or transmit
  - Configurable with standard (11-bit) or extended (29-bit) identifier
  - Supports programmable identifier receive mask
  - Supports data and remote frames
  - Holds 0 to 8 bytes of data
  - Parity-checked configuration and data RAM
- Individual identifier mask for each message object
- Programmable FIFO mode for message objects
- Programmable loop-back modes for self-test operation
- Suspend mode for debug support
- Software module reset
- Automatic bus-on, after bus-off state by a programmable 32-bit timer
- Message-RAM parity-check mechanism
- Two interrupt lines
- DMA support

---

**NOTE:** For a CAN bit clock of 100 MHz, the smallest bit rate possible is 3.90625 kbps.

---

---

**NOTE:** Depending on the timing settings used, the accuracy of the on-chip zero-pin oscillator (specified in the data manual) may not meet the requirements of the CAN protocol. In this situation, an external clock source must be used.

---

### 26.1.2 Functional Description

The CAN module performs CAN protocol communication according to ISO 11898-1. The bit rate can be programmed to values up to 1 Mbps. A CAN transceiver chip is required for the connection to the physical layer (CAN bus).

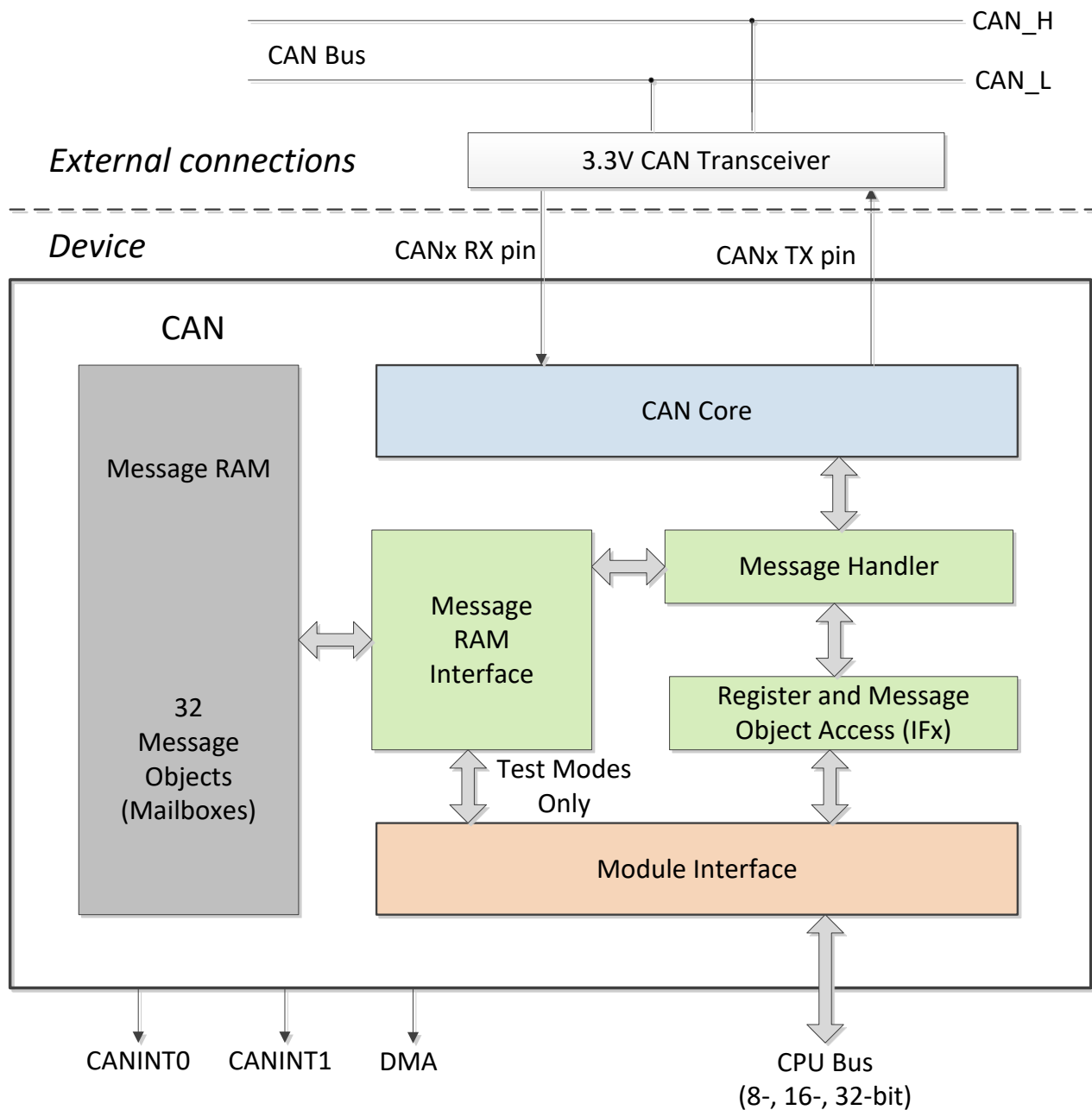
For communication on a CAN network, individual message objects can be configured. The message objects and identifier masks are stored in the Message RAM.

All functions concerning the handling of messages are implemented in the message handler. These functions are: acceptance filtering; the transfer of messages between the CAN Core and the Message RAM; and the handling of transmission requests as well as the generation of interrupts or DMA requests.

The register set of the CAN may be accessed directly by the CPU through the module interface. These registers are used to control and configure the CAN core and the message handler, and to access the message RAM.

### 26.1.3 Block Diagram

Figure 26-1. CAN Block Diagram



#### 26.1.3.1 CAN Core

The CAN core consists of the CAN Protocol Controller and the Rx/Tx Shift register. It handles all ISO 11898-1 protocol functions.

#### 26.1.3.2 Message Handler

The message handler is a state machine which controls the data transfer between the single-port Message RAM and the CAN Core's Rx/Tx Shift register. It also handles acceptance filtering and the interrupt request generation as programmed in the control registers.

### 26.1.3.3 Message RAM

The CAN message RAM enables the storage of 32 CAN messages.

### 26.1.3.4 Registers and Message Object Access (IFx)

Data consistency is ensured by indirect accesses to the message objects. During normal operation, all CPU and DMA accesses to the message RAM are done through Interface registers.

Three Interface register sets control the CPU read and write accesses to the Message RAM. There are two Interface register sets for read/write access (IF1 and IF2) and one Interface register set for read access only (IF3). See also [Section 26.14](#). The Interface registers have the same word length as the message RAM.

In a dedicated test mode, the message RAM is memory-mapped and can be directly accessed.

## 26.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some I/O functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pull-ups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

## 26.3 Address/Data Bus Bridge

The CAN module uses a special addressing scheme to support byte accesses. It is recommended to only use 32-bit accesses to the CAN registers using the *HWREG\_BP()* macro which uses the *\_\_byte\_peripheral\_32()* intrinsic. If 16-bit accesses are to be used, the lower 16 bits should be written to the register's address, and the upper 16 bits should be written to the register's address, plus 2.

Because of the bus bridge, the view of the CAN module's register space through a Code Composer Studio (CCS) memory window does not always match the actual addressing. When the view mode is 32-bit or 16-bit, even addresses are effectively duplicated; odd addresses should be ignored. When the view mode is 8-bit, even addresses from within the CAN module are duplicated into the odd addresses in the CCS memory view.; odd addresses from the module are not displayed.

**Table 26-1. CAN Register Access From Software**

CAN Register Space			C28x 8 Bit	
Address	Reg. Name	Data	Access	Data
0x00	CAN_CTL	0x33221100	__byte((int *)0x00,0)	0x0000
0x04	CAN_ES	0x77665544	__byte((int *)0x01,0)	0x0011
0x08	CAN_ERRC	0xBBAA9988	__byte((int *)0x02,0)	0x0022
0x0C	CAN_BTR	0xFFEEDDCC	__byte((int *)0x03,0)	0x0033
			__byte((int *)0x04,0)	0x0044
			__byte((int *)0x05,0)	0x0055
			__byte((int *)0x06,0)	0x0066
			__byte((int *)0x07,0)	0x0077
			__byte((int *)0x08,0)	0x0088
			__byte((int *)0x09,0)	0x0099
			__byte((int *)0x0A,0)	0x00AA
			__byte((int *)0x0B,0)	0x00BB
			__byte((int *)0x0C,0)	0x00CC
			__byte((int *)0x0D,0)	0x00DD
			__byte((int *)0x0E,0)	0x00EE
			__byte((int *)0x0F,0)	0x00FF
C28x 16 Bit		C28x 32 Bit		
Access	Data	Access	Data	
((short *)0x00))	0x1100	((long *)0x00))	0x33221100	
((short *)0x01))	0x1100	((long *)0x01))	0x33221100	
((short *)0x02))	0x3322	((long *)0x02))	0x33221100	
((short *)0x03))	0x3322	((long *)0x03))	0x33221100	
((short *)0x04))	0x5544	((long *)0x04))	0x77665544	
((short *)0x05))	0x5544	((long *)0x05))	0x77665544	
((short *)0x06))	0x7766	((long *)0x06))	0x77665544	
((short *)0x07))	0x7766	((long *)0x07))	0x77665544	
((short *)0x08))	0x9988	((long *)0x08))	0xBBAA9988	
((short *)0x09))	0x9988	((long *)0x09))	0xBBAA9988	
((short *)0x0A))	0xBBAA	((long *)0x0A))	0xBBAA9988	
((short *)0x0B))	0xBBAA	((long *)0x0B))	0xBBAA9988	
((short *)0x0C))	0xDDCC	((long *)0x0C))	0xFFEEDDCC	
((short *)0x0D))	0xDDCC	((long *)0x0D))	0xFFEEDDCC	
((short *)0x0E))	0xFFEE	((long *)0x0E))	0xFFEEDDCC	



**Table 26-1. CAN Register Access From Software (continued)**

CAN Register Space		C28x 8 Bit	
<code>((short *)0x0F)</code>	0xFFEE	<code>((long *)0x0F)</code>	0xFFEEDCC

**Table 26-2. CAN Register Access From CCS**

CCS 8 Bit		CCS 16 Bit		CCS 32 Bit	
Address	Displayed Data	Address	Displayed Data	Address	Displayed Data
0x00	0x00	0x00	0x1100	0x00	0x11001100
0x01	0x00	0x01	0x1100	0x02	0x33223322
0x02	0x22	0x02	0x3322	0x04	0x55445544
0x03	0x22	0x03	0x3322	0x06	0x77667766
0x04	0x44	0x04	0x5544	0x08	0x99889988
0x05	0x44	0x05	0x5544	0x0A	0xBBAABBAA
0x06	0x66	0x06	0x7766	0x0C	0xDDCCDDCC
0x07	0x66	0x07	0x7766	0x0E	0xFFEEFFEE
0x08	0x88	0x08	0x9988		
0x09	0x88	0x09	0x9988		
0x0A	0xAA	0x0A	0xBBAA		
0x0B	0xAA	0x0B	0xBBAA		
0x0C	0xCC	0x0C	0xDDCC		
0x0D	0xCC	0x0D	0xDDCC		
0x0E	0xEE	0x0E	0xFFEE		
0x0F	0xEE	0x0F	0xFFEE		

## 26.4 Operating Modes

### 26.4.1 Initialization

The initialization mode is entered either by software (by setting the **Init** bit in the CAN\_CTL register), by hardware reset, or by going bus-off. While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN\_TX output is recessive (high). The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

To initialize the CAN Controller, the CPU has to configure the CAN Bit timing and those message objects which have to be used for CAN communication. Message objects which are not needed, can be deactivated with their MsgVal bits cleared.

The access to the Bit Timing Register for the configuration of the bit timing is enabled when both **Init** and **CCE** bits in the CAN Control register are set.

Clearing the Init bit finishes the software initialization. Afterwards, the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus Idle) before it can take part in bus activities and start the message transfer. For more details see [Section 26.13](#).

The initialization of the message objects is independent of the Init bit; however, all message objects should be configured to particular identifiers or set to "not-valid" before the message transfer is started.

It is possible to change the configuration of message objects during normal operation by the CPU. After setup and subsequent transfer of message object from interface registers to message RAM, the acceptance filtering will be applied to it when the modified message object number is same or smaller than the previously found message object. This assures data consistency even when changing message objects; for example, while there is a pending CAN frame reception.

### 26.4.2 CAN Message Transfer (Normal Operation)

Once the CAN is initialized and the Init bit is reset to zero, the CAN Core synchronizes itself to the CAN bus and is ready for communication.

Received messages are stored into their appropriate message objects if they pass acceptance filtering. The whole message (MSGID, DLC, and up to eight data bytes) is stored into the message object. As a consequence, for example, if the identifier mask is used, the MSGID bits which are masked to "don't care" may change in the message object when a received message is stored.

The CPU may read or write each message at any time via the Interface registers, as the message handler guarantees data consistency in case of concurrent accesses.

Messages to be transmitted can be updated by the CPU. If a permanent message object (MSGID, control bits set up during configuration, and setup for multiple CAN transfers) exists for the message, it is possible to only update the data bytes. If several transmit messages should be assigned to one message object, the whole message object has to be configured before the transmission of this message is requested.

The transmission of multiple message objects may be requested at the same time. They are subsequently transmitted, according to their internal priority. Messages may be updated or set to 'not valid' at any time, even if a requested transmission is still pending. However, the data bytes will be discarded if a message is updated before a pending transmission has started.

Depending on the configuration of the message object, a transmission may be automatically requested by the reception of a remote frame with a matching identifier.

#### 26.4.2.1 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the CAN provides a mechanism to automatically retransmit frames which have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed

By default, this automatic retransmission is enabled. It can be disabled by setting the DAR bit in the CAN Control register. Further details to this mode are provided in [Section 26.12.3](#).

#### 26.4.2.2 Auto-Bus-On

After the CAN has entered the bus-off state, the CPU can start a bus-off-recovery sequence by resetting the *Init* bit. If this is not done, the module will stay in bus-off state.

The CAN provides an automatic auto-bus-on feature which is enabled by the ABO bit. If set, the CAN will automatically start the bus-off-recovery sequence. The sequence can be delayed by a user-defined number of clock cycles.

---

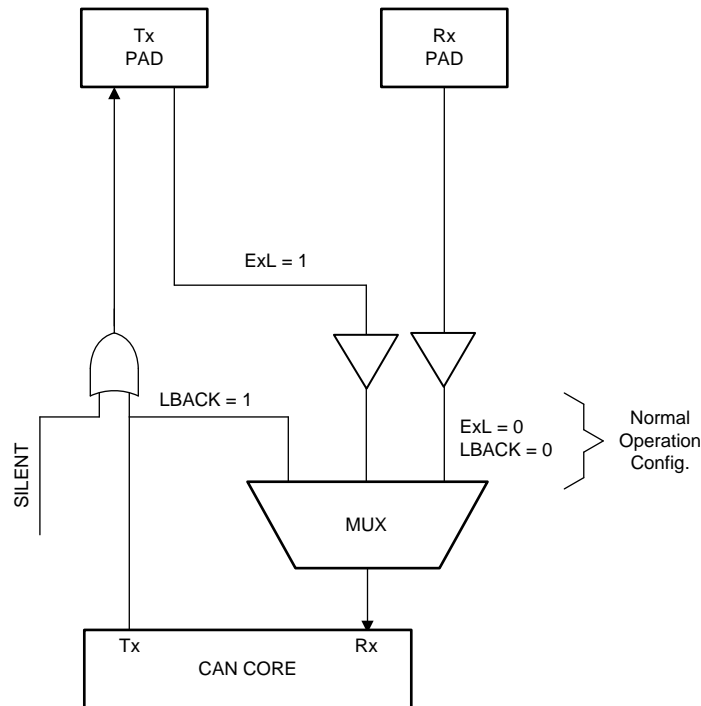
**NOTE:** If the CAN module goes Bus-Off due to multiple CAN bus errors, it stops all bus activities and automatically sets the *Init* bit. Once the *Init* bit is cleared by the application (or due to the auto-bus-on feature), the device will wait for 129 occurrences of Bus Idle (equal to  $129 * 11$  consecutive recessive bits) before resuming normal operation. The Bus-Off recovery sequence cannot be shortened by setting or resetting the *Init* bit. At the end of the bus-off recovery sequence, the error counters will be reset. After the *Init* bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 Error code is written to the Error and Status Register. This enables the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the Bus-Off recovery sequence.

---

#### 26.4.3 Test Modes

The CAN module provides several test modes which are mainly intended for self-test purposes. The diagram below aids in understanding the various test modes. It should be viewed as representative of the module behavior, and not as a gate-accurate implementation of the module. For the sake of brevity, this diagram does not include the GPIO muxing or the I/O buffers.

Figure 26-2. CAN\_MUX



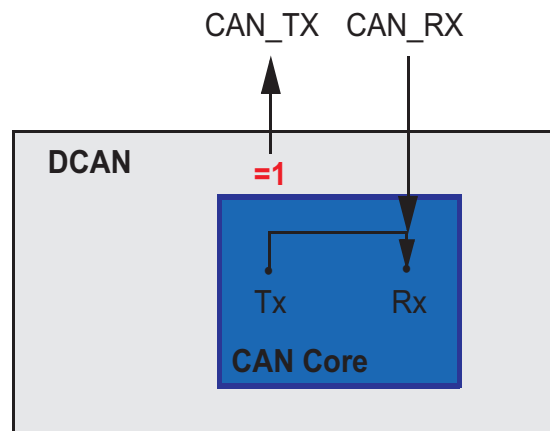
For all test modes, the Test bit in the CAN Control register needs to be set to 1 to enable write access to the CAN\_TEST register.

### 26.4.3.1 Silent Mode

The silent mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (for example, acknowledge bit, overload flag, active error flag). The CAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, the received frames are internally routed to the CAN Core.

Figure 26-3 shows the connection of signals CAN\_TX and CAN\_RX to the CAN core in silent mode. Silent mode can be activated by setting the **Silent** bit in test register (CAN\_TEST), to 1. In ISO 11898-1, the silent mode is called the bus monitoring mode.

Figure 26-3. CAN Core in Silent Mode



### 26.4.3.2 Loopback Mode

The loopback mode is mainly intended for hardware self-test functions. In this mode, the CAN core uses internal feedback from Tx output to Rx input. Transmitted messages are treated as received messages, and can be stored into message objects if they pass acceptance filtering. The actual value of the CAN\_RX input pin is disregarded by the CAN core. Transmitted messages still can be monitored at the CAN\_TX pin.

In order to be independent from external stimulation, the CAN core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loopback mode.

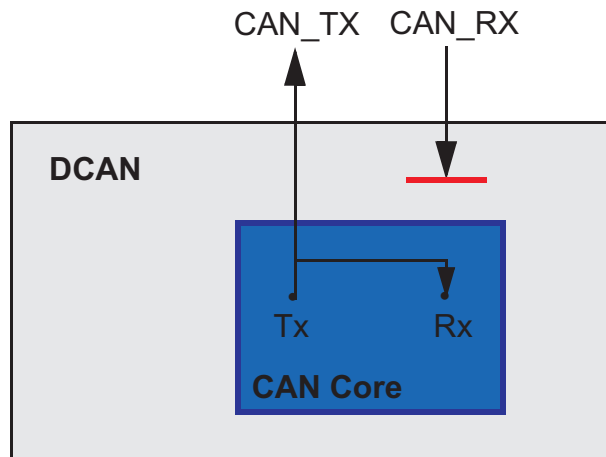
Figure 26-4 shows the connection of signals CAN\_TX and CAN\_RX to the CAN core in loopback mode. Loopback mode can be activated by setting bit **LBack** in the CAN\_TEST register to 1.

---

**NOTE:** In loopback mode, the signal path from CAN core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN Core are disregarded. For including these into the testing, see Section 26.4.3.3.

---

**Figure 26-4. CAN Core in Loopback Mode**



### 26.4.3.3 External Loopback Mode

The external loopback mode is similar to the loopback mode; however, it includes the signal path from CAN Core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to the CAN Core. When the external loopback mode is selected, the CAN core is connected to the input buffer of the Tx pin. With this configuration, the Tx pin IO circuit can be tested. External loopback mode can be activated by setting the **ExL** bit in Test Register to 1.

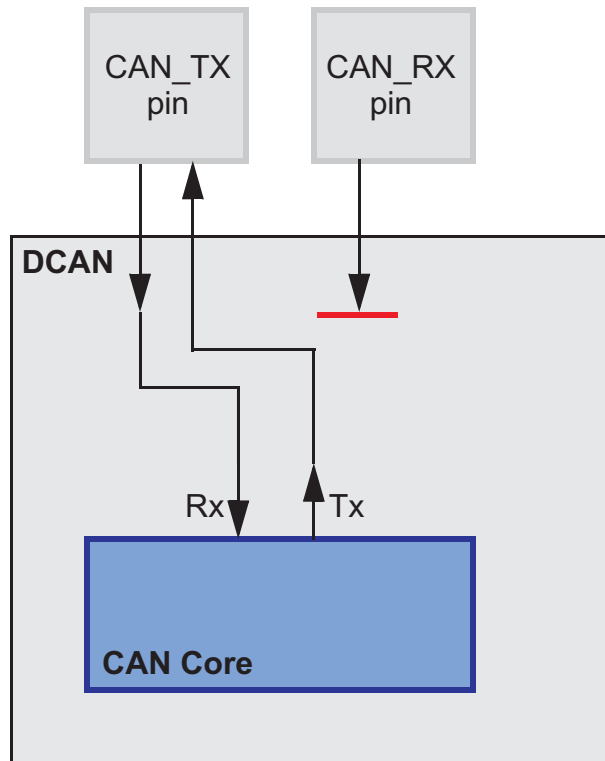
Figure 26-5 shows the connection of signals CAN\_TX and CAN\_RX to the CAN Core in external loopback mode.

---

**NOTE:** When loopback mode is active (LBack bit set), the ExL bit will be ignored.

---

Figure 26-5. CAN Core in External Loopback Mode

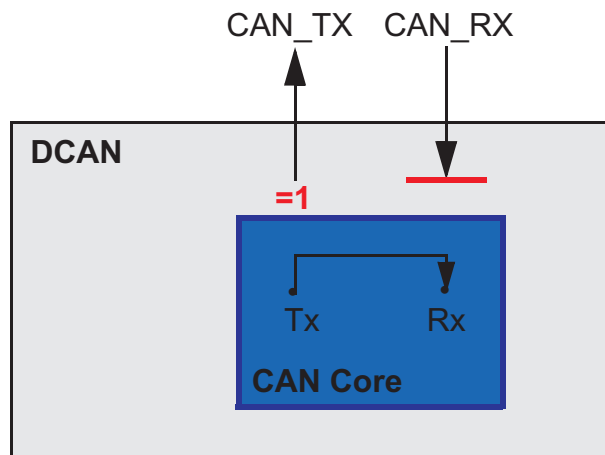


#### 26.4.3.4 Loopback Combined with Silent Mode

It is also possible to combine loopback mode and silent mode by setting bits LBack and Silent at the same time. The CAN hardware can be tested without affecting the CAN network. In this mode, the CAN\_RX pin is disconnected from the CAN Core and no dominant bits will be sent on the CAN\_TX pin.

Figure 26-6 shows the connection of the signals CAN\_TX and CAN\_RX to the CAN Core in case of the combination of loopback mode with silent mode.

Figure 26-6. CAN Core in Loopback Combined with Silent Mode



## 26.5 Multiple Clock Source

The CAN bit timing clock is normally derived from the system clock (SYSCLK). If desired, the bit clock can be derived directly from the external clock source (XTAL).

The System Control chapter of this document and the device data manual provide for more information on how to configure the relevant clock source registers in the system module.

---

**NOTE:** The CAN core has to be programmed to at least eight clock cycles per bit time. To achieve a transfer rate of 1 Mbps an oscillator frequency of 8 MHz or higher has to be used.

---

## 26.6 Interrupt Functionality

Interrupts can be generated on two interrupt lines: CAN0INT and CAN1INT. These lines can be enabled by setting the IE0 and IE1 bits, respectively, in the CAN Control register.

The CAN provides three groups of interrupt sources: message object interrupts, status change interrupts and error interrupts. The source of an interrupt can be determined by the interrupt identifiers Int0ID / Int1ID in the Interrupt register. When no interrupt is pending, the register will hold the value zero. Each interrupt line remains active until the dedicated field in the Interrupt register (Int0ID / Int1ID) again reach zero (this means the cause of the interrupt is reset), or until IE0 / IE1 are reset. The value 0x8000 in the Int0ID field indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Error and Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits RxOk, TxOk and LEC by reading the Error and Status Register, but a write access of the CPU will never generate or reset an interrupt.

Values between 1 and the number of the last message object indicates that the source of the interrupt is one of the message objects. INT0ID / INT1ID will point to the pending message interrupt with the highest priority. The Message Object 1 has the highest priority, the last message object has the lowest priority.

An interrupt service routine which reads the message from the interrupt source, may also read the message and reset the message object's IntPnd at the same time (ClrIntPnd bit in the IF1/IF2 Command register). When IntPnd is cleared, the Interrupt register will point to the next message object with a pending interrupt.

The CAN module features a module-level interrupt enable and acknowledge mechanism. To enable the CAN0 and CAN1 interrupts, you must set the appropriate bits in the CAN\_GLB\_INT\_EN register. When handling an interrupt, the individual message or status change flag must be cleared prior to acknowledging the interrupt via CAN\_GLB\_INT\_CLR and PIEACK.

### 26.6.1 Message Object Interrupts

Message object interrupts are generated by events from the message objects. They are controlled by the flags IntPND, TxIE and RxIE which are described in [Section 26.15.1](#). Message object interrupts can be routed to the CAN0INT or CAN1INT line, which is controlled by the Interrupt Multiplexer register.

### 26.6.2 Status Change Interrupts

The events RxOk, TxOk, and LEC in the Error and Status register belong to the status change interrupts. The status change interrupt group can be enabled by the SIE bit in the CAN Control Register. If SIE is set, a status change interrupt will be generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the Message RAM configuration. Status Change interrupts can only be routed to interrupt line CAN0INT which has to be enabled by setting IE0 in the CAN\_CTL Register.

### 26.6.3 Error Interrupts

The events PER, BOff and EWarn, belong to the error interrupts. The error interrupt group can be enabled by setting bit EIE. Also, error interrupts can only be routed to interrupt line CAN0INT which has to be enabled by setting IE0 in the CAN\_CTL register.

### 26.6.4 PIE Nomenclature for DCAN Interrupts

[Table 26-3](#) shows the PIE nomenclature for the interrupts.

**Table 26-3. PIE Nomenclature for Interrupts**

Interrupt	CANA	CANB
CANINT0	CANA_0	CANB_0
CANINT1	CANA_1	CANB_1

### 26.6.5 Interrupt Topologies

Interrupt topologies for CAN are illustrated in [Figure 26-7](#) and [Figure 26-8](#).

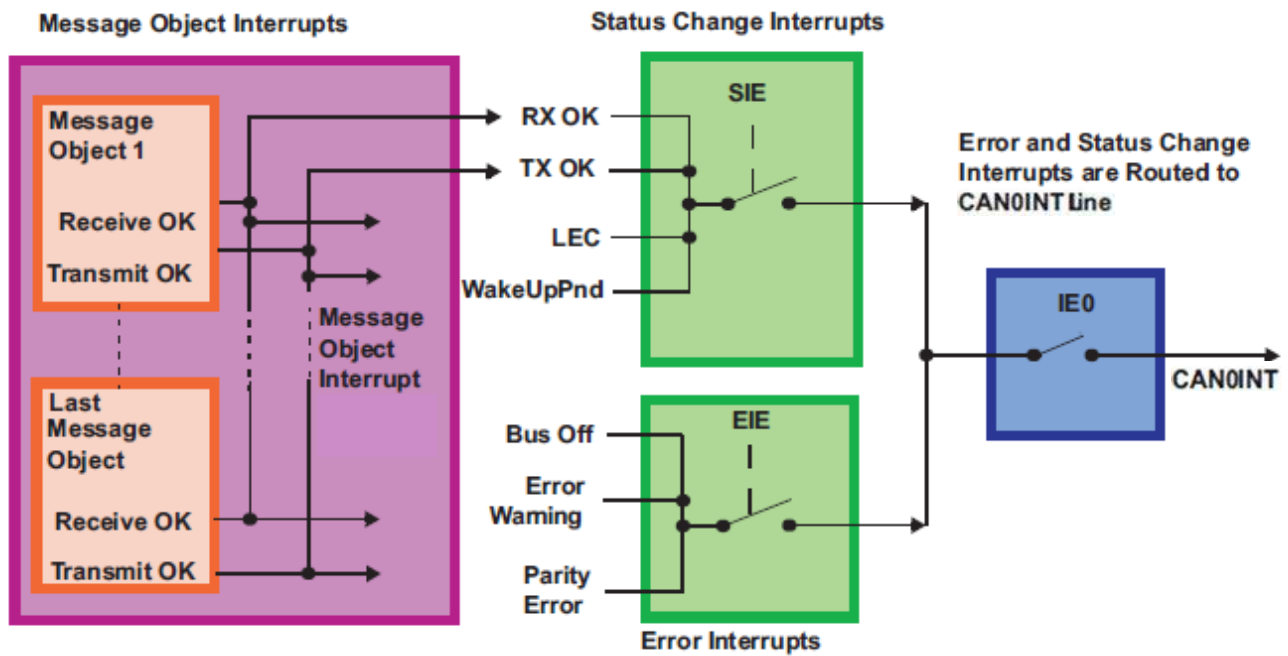
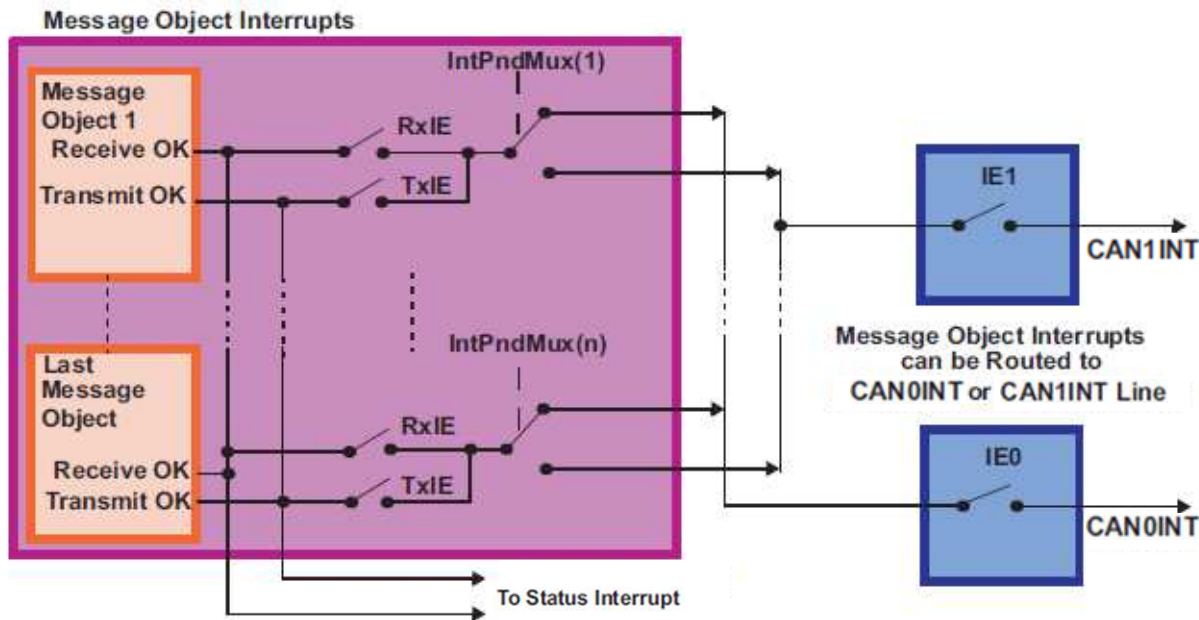
**Figure 26-7. CAN Interrupt Topology 1**




Figure 26-8. CAN Interrupt Topology 2



### 26.7 DMA Functionality

The CAN module provides three DMA trigger outputs, one for each of the three Interface Registers IF1, IF2 and IF3. These can be enabled using the DE1, DE2, and DE3 bits in the CAN\_CTL register.

The Update of IF1 and IF2 registers will be initiated by a write access to the IF1 and IF2 Command registers, respectively. Once enabled, setting the DMAActive bit in the IF1CMD or IF2CMD registers will cause a DMA request the next time the corresponding interface becomes available.

The IF3 registers content can be automatically updated on reception of CAN messages in message objects which are programmed for automatic IF3 update. That is, when IF3 DMA requests are enabled, all IF3 updates will trigger a DMA request.

When a DCAN internal IFx update is complete, a DMA request will be activated and stays active until the first access to one of the relevant IFx registers; that is, DMA requests are cleared after the first read or write access to an IF register set.

### 26.8 Parity Check Mechanism

The CAN provides a parity check mechanism to ensure data integrity of message RAM data. For each word (32 bits) in Message RAM, one parity bit will be calculated.

Parity information is stored in the Message RAM on write accesses and will be checked against the stored parity bit from Message RAM on read accesses.

The parity check functionality can be enabled or disabled by the PMD bit field in the CAN Control register. In case of a disabled parity check, the parity bits in message RAM will be left unchanged on write access to data area and no check will be done on read access.

If parity checking is enabled, parity bits will be automatically generated and checked by the CAN. A parity bit will be set if the modulo-2-sum of the data bits is 1. This means that if the parity bit is set, then there are an odd number of 1 bits in the data.

### 26.8.1 Behavior on Parity Error

On any read access to Message RAM, for example, during the start of a CAN frame transmission, the parity of the message object will be checked. If a parity error is detected, the PER bit in the Error and Status register will be set. If error interrupts are enabled, an interrupt would also be generated. In order to avoid the transmission of invalid data over the CAN bus, the MsgVal bit of the message object will be reset.

The message object data can be read by the CPU, independently of parity errors. Thus, the application has to ensure that the read data is valid, for example, by immediately checking the Parity Error Code register on parity error interrupt.

## 26.9 Debug Mode

The module supports the usage of an external debug unit by providing functions like pausing CAN activities and making message RAM content accessible from the debugger. Debug mode is entered automatically when an external debugger is connected and the core is halted.

Before entering Debug mode, the circuit will wait until a transmission is started, a reception is finished, or the Bus idle state is recognized. If the IDS bit is set, the debugger immediately interrupts the current transmission or reception. Afterwards, the CAN enters Debug mode, indicated by the InitDbg flag, in the CAN Control register. During debug mode, all CAN registers can be accessed. Reading reserved bits will return '0'. Writing to reserved bits will have no effect. Also, the message RAM will be memory mapped. This allows the external debug unit to read the message RAM. For the memory organization (see [Section 26.15.3](#)).

---

**NOTE:** During debug mode, the Message RAM cannot be accessed via the IFx register sets.

---



---

**NOTE:** Writing to control registers in Debug mode may influence the CAN state machine and further message handling.

---

For debug support, the auto clear functionality of the following CAN registers is disabled:

- Error and Status register (clear of status flags by read)
- IF1/IF2 Command registers (clear of DMA Active flag by r/w)

## 26.10 Module Initialization

After hardware reset, the Init bit in the CAN Control register is set and all CAN protocol functions are disabled. The configuration of the bit timing and of the message objects should be completed before the CAN protocol functions are enabled.

For the configuration of the message objects, see [Section 26.11](#).

For the configuration of the Bit Timing, see [Section 26.13.2](#).

The bits MsgVal, NewDat, IntPnd, and TxRqst of the message objects are reset to '0' by a hardware reset. The configuration of a message object is done by programming Mask, Arbitration, Control and Data bits of one of the IF1/IF2 Interface register sets to the desired values. By writing the message object number to bits [7:0] of the corresponding IF1/IF2 Command register, the IF1/IF2 Interface Register content is loaded into the addressed message object in the Message RAM.

The configuration of the bit timing requires that the CCE bit in the CAN Control register is set additionally to Init. This is not required for the configuration of the message objects.

When the Init bit in the CAN Control register is cleared, the CAN Protocol Controller state machine of the CAN Core and the message handler State Machine start to control the CAN's internal data flow. Received messages which pass the acceptance filtering are stored into the Message RAM; messages with pending transmission request are loaded into the CAN Core's Shift register and are transmitted via the CAN bus.

The CPU may enable the interrupt lines (setting IE0 and IE1 to '1') at the same time when it clears Init and CCE. The status interrupts EIE and SIE may be enabled simultaneously.

The CAN communication may be controlled interrupt-driven or in polling mode. The Interrupt Register points to those message objects with `IntPnd = '1'`. It is updated even if the interrupt lines to the CPU are disabled (`IE0 / IE1` are zero).

The CPU may poll all MessageObject's `NewDat` and `TxRqst` bits in parallel from the `NewData` registers and the `Transmission Request` registers. Polling can be made easier if all `Transmit Objects` are grouped at the low numbers; all `Receive Objects` are grouped at the high numbers.

## 26.11 Configuration of Message Objects

The entire Message RAM should to be configured before the end of the initialization; however, it is also possible to change the configuration of message objects during CAN communication.

### 26.11.1 Configuration of a Transmit Object for Data Frames

Figure 26-9 shows how a transmit object can be initialized.

**Figure 26-9. Initialization of a Transmit Object**

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

- The arbitration bits (`ID[28:0]` and `Xtd` bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (standard frame) is used (`Xtd = '0'`), it is programmed to `ID[28:18]`. In this case, `ID[17:0]` can be ignored.
- The data registers (`DLC[3:0]` and `Data0-7`) are given by the application. `TxRqst` and `RmtEn` should not be set before the data is valid.
- If the `TxIE` bit is set, the `IntPnd` bit will be set after a successful transmission of the message object.
- If the `RmtEn` bit is set, a matching received remote frame will cause the `TxRqst` bit to be set; the remote frame will autonomously be answered by a data frame.
- The Mask bits (`Msk[28:0]`, `UMask`, `MXtd`, and `MDir` bits) may be used (`UMask='1'`) to allow groups of remote frames with similar identifiers to set the `TxRqst` bit. The `Dir` bit should not be masked. For details see [Section 26.12.8](#). Identifier masking must be disabled (`UMask = '0'`) if no remote frames are allowed to set the `TxRqst` bit (`RmtEn = '0'`).

### 26.11.2 Configuration of a Transmit Object for Remote Frames

It is not necessary to configure transmit objects for the transmission of remote frames. Setting `TxRqst` for a receive object will cause the transmission of a remote frame with the same identifier as the data frame for which this receive object is configured.

### 26.11.3 Configuration of a Single Receive Object for Data Frames

Figure 26-10 shows how a receive object for data frames can be initialized.

**Figure 26-10. Initialization of a Single Receive Object for Data Frames**

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

- The arbitration bits (`ID[28:0]` and `Xtd` bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (Standard Frame) is used (`Xtd = '0'`), it is programmed to `ID[28:18]`. In this case, `ID[17:0]` can be ignored. When a data frame with an 11-bit Identifier is received, `ID[17:0]` will be set to '0'.
- When the message handler stores a data frame in the message object, it will store the received data length code and the corresponding number of data bytes. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by non specified values.
- The mask bits (`Msk[28:0]`, `UMask`, `MXtd`, and `MDir` bits) may be used (`UMask = '1'`) to allow groups of data frames with similar identifiers to be accepted. The `Dir` bit should not be masked in typical applications. If some bits of the Mask bits are set to "don't care", the corresponding bits of the Arbitration Register will be overwritten by the bits of the stored data frame.

- If the RxIE bit is set, the IntPnd bit will be set when a received data frame is accepted and stored in the message object.
- If the TxRqst bit is set, the transmission of a remote frame with the same identifier as actually stored in the Arbitration bits will be triggered. The content of the Arbitration bits may change if the Mask bits are used (UMask = '1') for acceptance filtering.

### 26.11.4 Configuration of a Single Receive Object for Remote Frames

Figure 26-11 shows how a receive object for remote frames can be initialized.

**Figure 26-11. Initialization of a single Receive Object for Remote Frames**

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	appl.	0	0	0	0

- Receive objects for remote frames may be used to monitor remote frames on the CAN bus. The remote frame stored in the receive object will not trigger the transmission of a data frame. Receive objects for remote frames may be expanded to a FIFO buffer, see [Section 26.11.5](#).
- UMask must be set to '1'. The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be set to "must-match" or to "don't care", to allow groups of remote frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. For details see [Section 26.12.8](#).
- The arbitration bits (ID[28:0] and Xtd bit) may be given by the application. They define the identifier and type of accepted received remote frames. If some bits of the Mask bits are set to "don't care", the corresponding bits of the arbitration bits will be overwritten by the bits of the stored remote frame. If an 11-bit Identifier (standard frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a remote frame with an 11-bit Identifier is received, ID[17:0] will be set to '0'.
- The data length code (DLC[3:0]) may be given by the application. When the message handler stores a remote frame in the message object, it will store the received data length code. The data bytes of the message object will remain unchanged.
- If the RxIE bit is set, the IntPnd bit will be set when a received remote frame is accepted and stored in the message object.

### 26.11.5 Configuration of a FIFO Buffer

With the exception of the EoB bit, the configuration of receive objects belonging to a FIFO buffer is the same as the configuration of a single receive object.

To concatenate multiple message objects to a FIFO buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest number will be the first message object of the FIFO buffer. The EoB bit of all message objects of a FIFO buffer except the last one must be programmed to zero. The EoB bits of the last message object of a FIFO buffer is set to one, configuring it as the end of the block.

## 26.12 Message Handling

When initialization is finished, the CAN module synchronizes itself to the traffic on the CAN bus. It does acceptance filtering on received messages and stores those frames that are accepted into the designated message objects. The application must update the data of the messages to be transmitted and to enable and request their transmission. The transmission is requested automatically when a matching remote frame is received.

The application may read messages which are received and accepted. Messages that are not read before the next messages are accepted for the same message object will be overwritten. Messages may be read interrupt-driven or after polling of NewDat.

### 26.12.1 Message Handler Overview

The message handler state machine controls the data transfer between the Rx/Tx Shift Register of the CAN Core and the Message RAM. It performs the following tasks:

- Data transfer from Message RAM to CAN Core (messages to be transmitted).

- Data transfer from CAN Core to the Message RAM (received messages).
- Data transfer from CAN Core to the Acceptance Filtering unit.
- Scanning of Message RAM for a matching message object (acceptance filtering).
- Scanning the same message object after being changed by IF1/IF2 registers when priority is same or higher as message the object found by last scanning.
- Handling of TxRqst flags.
- Handling of interrupt flags.

The message handler registers contains status flags of all message objects grouped into the following topics:

- Transmission request flags
- New data flags
- Interrupt pending flags
- Message valid registers

Instead of collecting above listed status information of each message object via IFx registers separately, these message handler registers provide a fast and easy way to get an overview, for example, about all pending transmission requests.

All message handler registers are read-only.

### 26.12.2 Receive/Transmit Priority

The receive/transmit priority for the message objects is attached to the message number, not to the CAN identifier. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding message object, so for example, messages with the highest priority can be placed in the message objects with the lowest numbers.

The acceptance filtering for received data frames or remote frames is also done in ascending order of message objects, so a frame that has been accepted by a message object cannot be accepted by another message object with a higher message number. The last message object may be configured to accept any data frame or remote frame that was not accepted by any other message object, for nodes that need to log the complete message traffic on the CAN bus.

### 26.12.3 Transmission of Messages in Event Driven CAN Communication

If the shift register of the CAN Core is ready for loading and if there is no data transfer between the IFx registers and Message RAM, the MsgVal bits in the Message Valid register and the TxRqst bits in the transmission request register are evaluated. The valid message object with the highest priority pending transmission request is loaded into the shift register by the message handler and the transmission is started. The message object's NewDat bit is reset.

After a successful transmission and if no new data was written to the message object (NewDat = '0') since the start of the transmission, the TxRqst bit will be reset. If TxIE is set, IntPnd will be set after a successful transmission. If the CAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

If automatic retransmission mode is disabled by setting the DAR bit in the CAN Control register, the behavior of bits TxRqst and NewDat in the Message Control register of the Interface register set is as follows:

- When a transmission starts, the TxRqst bit of the respective Interface register set is reset, while bit NewDat remains set.
- When the transmission has been successfully completed, the NewDat bit is reset.

When a transmission failed (lost arbitration or error) bit NewDat remains set. To restart the transmission, the application has to set TxRqst again.



Received remote frames do not require a receive object. They will automatically trigger the transmission of a data frame, if in the matching Transmit Object the RmtEn bit is set.

#### 26.12.4 Updating a Transmit Object

The CPU may update the data bytes of a transmit object any time via the IF1/IF2 interface registers, neither MsgVal nor TxRqst have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes in the corresponding IF1/IF2 Data A register or IF1/IF2 Data B register have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the IF1/IF2 Data register or the message object is transferred to the IF1/IF2 Data Register before the CPU writes the new data bytes.

When only the data bytes are updated, first 0x87 can be written to bits [23:16] of the Command register and then the number of the message object is written to bits [7:0] of the Command register, concurrently updating the data bytes and setting TxRqst with NewDat.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat has to be set together with TxRqst in event driven CAN communication. For details see [Section 26.12.3](#).

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

#### 26.12.5 Changing a Transmit Object

If the number of implemented message objects is not sufficient to be used as permanent message objects only, the transmit objects may be managed dynamically. The CPU can write the whole message (arbitration, control, and data) into the Interface register. The bits [23:16] of the Command register can be set to 0xB7 for the transfer of the whole message object content into the message object. Neither MsgVal nor TxRqst have to be reset before this operation.

If a previously requested transmission of this message object is not completed but already in progress, it will be continued; however it will not be repeated if it is disturbed.

To update only the data bytes of a message being transmitted, set bits [23:16] of the Command register to 0x87.

---

**NOTE:** After the update of the transmit object, the interface register set will contain a copy of the actual contents of the object, including the part that had not been updated.

---

#### 26.12.6 Acceptance Filtering of Received Messages

When the arbitration and control bits (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the shift register of the CAN Core, the message handler starts to scan the message RAM for a matching valid message object:

- The acceptance filtering unit is loaded with the arbitration bits from the CAN Core shift register.
- Then the arbitration and mask bits (including MsgVal, UMask, NewDat, and EoB) of Message Object 1 are loaded into the Acceptance Filtering unit and are compared with the arbitration bits from the shift register. This is repeated for all following message objects until a matching message object is found, or until the end of the Message RAM is reached.
- If a match occurs, the scanning is stopped and the message handler proceeds depending on the type of the frame (data frame or remote frame) received.

#### 26.12.7 Reception of Data Frames

The message handler stores the message from the CAN Core shift register into the respective message object in the Message RAM. Not only the data bytes, but all arbitration bits and the data length code are stored into the corresponding message object. This ensures that the data bytes stay associated to the identifier even if arbitration mask registers are used.

The NewDat bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset the NewDat bit when it reads the message object. If at the time of the reception the NewDat bit was already set, MsgLst is set to indicate that the previous data (not seen by the CPU) is lost. If the RxIE bit is set, the IntPnd bit is set, causing the Interrupt Register to point to this message object.

The TxRqst bit of this message object is reset to prevent the transmission of a remote frame, while the requested data frame has just been received.

### **26.12.8 Reception of Remote Frames**

When a remote frame is received, three different configurations of the matching message object have to be considered:

1. Dir = '1' (direction = transmit), RmtEn = '1', UMask = '1' or '0'

The TxRqst bit of this message object is set at the reception of a matching remote frame. The rest of the message object remains unchanged.

2. Dir = '1' (direction = transmit), RmtEn = '0', UMask = '0'

The remote frame is ignored, this message object remains unchanged.

3. Dir = '1' (direction = transmit), RmtEn = '0', UMask = '1'

The remote frame is treated similar to a received data frame. At the reception of a matching remote frame, the TxRqst bit of this message object is reset. The arbitration and control bits (Identifier + IDE + RTR + DLC) from the shift register are stored in the message object in the Message RAM and the NewDat bit of this message object is set. The data bytes of the message object remain unchanged.

### **26.12.9 Reading Received Messages**

The CPU may read a received message any time via the IFx interface registers, the data consistency is guaranteed by the message handler state machine.

Typically the CPU will write 0x7F to bits [23:16] and then the number of the message object to bits [7:0] of the Command Register. That combination will transfer the whole received message from the Message RAM into the Interface Register set. Additionally, the bits NewDat and IntPnd are cleared in the Message RAM (not in the Interface Register set). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

If the message object uses masks for acceptance filtering, the arbitration bits show which of the different matching messages has been received.

The actual value of NewDat shows whether a new message has been received since last time when this message object was read. The actual value of MsgLst shows whether more than one message have been received since the last time when this message object was read. MsgLst will not be automatically reset.

### **26.12.10 Requesting New Data for a Receive Object**

By means of a remote frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TxRqst bit of a receive object will cause the transmission of a remote frame with the receive object's identifier. This remote frame triggers the other CAN node to start the transmission of the matching data frame. If the matching data frame is received before the remote frame could be transmitted, the TxRqst bit is automatically reset.

Setting the TxRqst bit without changing the contents of a message object requires the value 0x84 in bits [23:16] of the Command Register.

### **26.12.11 Storing Received Messages in FIFO Buffers**

Several message objects may be grouped to form one or more FIFO Buffers. Each FIFO Buffer configured to store received messages with a particular (group of) Identifier(s). Arbitration and Mask registers of the FIFO Buffer's message objects are identical. The EoB (End of Buffer) bits of all but the last of the FIFO Buffer's message objects are '0', in the last one the EoB bit is '1'.

Received messages with identifiers matching to a FIFO Buffer are stored into a message object of this FIFO Buffer, starting with the message object with the lowest message number.

When a message is stored into a message object of a FIFO Buffer the NewDat bit of this message object is set. By setting NewDat while EoB is '0' the message object is locked for further write accesses by the message handler until the CPU has cleared the NewDat bit.

Messages are stored into a FIFO Buffer until the last message object of this FIFO Buffer is reached. If none of the preceding message objects is released by writing NewDat to '0', all further messages for this FIFO Buffer will be written into the last message object of the FIFO Buffer (EoB = '1') and therefore overwrite previous messages in this message object.

### 26.12.12 Reading from a FIFO Buffer

Several messages may be accumulated in a set of message objects which are concatenated to form a FIFO Buffer before the application program is required (in order to avoid the loss of data) to empty the buffer. A FIFO Buffer of length N will store N-1 plus the last received message since last time it was cleared. A FIFO Buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in [Figure 26-12](#).

---

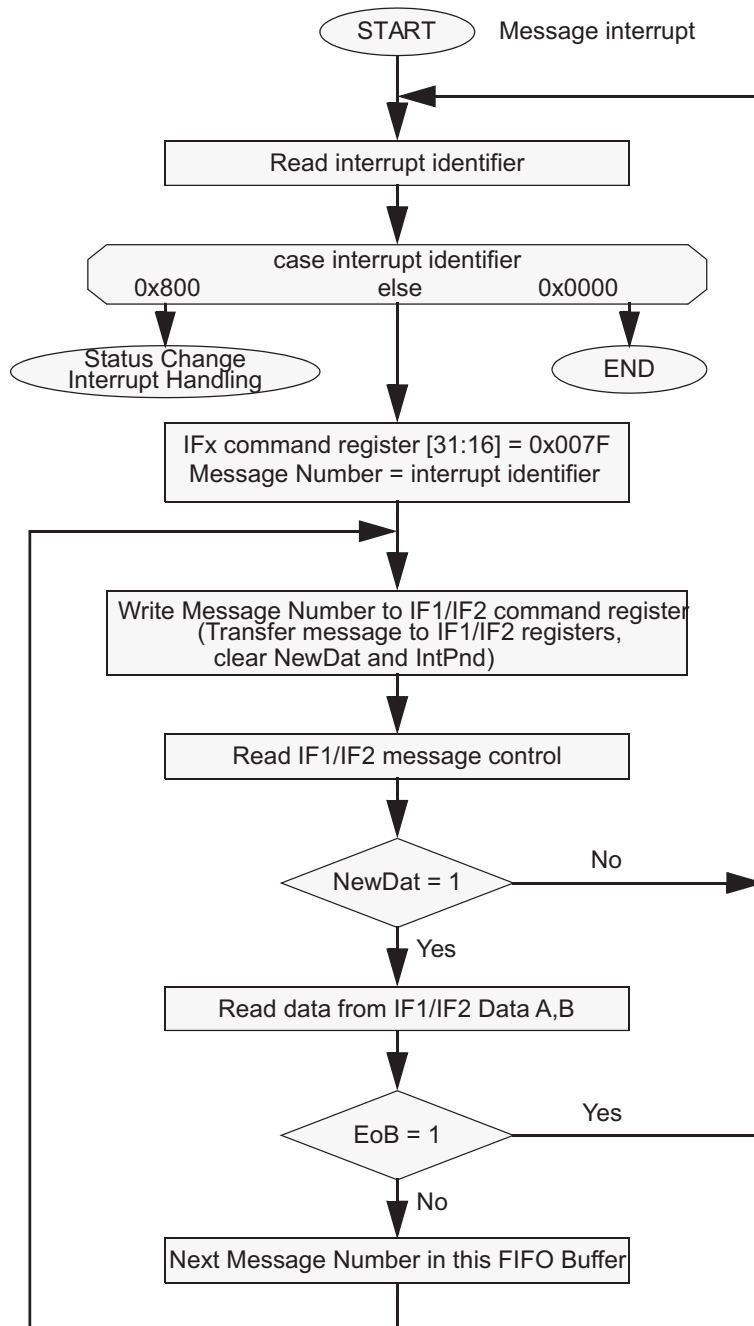
**NOTE:** All message objects of a FIFO buffer needs to be read and cleared before the next batch of messages can be stored. Otherwise true FIFO functionality cannot be guaranteed, since the message objects of a partly read buffer will be re-filled according to the normal (descending) priority.

---

Reading from a FIFO Buffer message object and resetting its NewDat bit is handled the same way as reading from a single message object.



Figure 26-12. CPU Handling of a FIFO Buffer (Interrupt Driven)



### 26.13 CAN Bit Timing

The CAN supports bit rates between less than 1 kBit/s and 1000 kBit/s.

Each member of the CAN network has its own clock generator, typically derived from a crystal oscillator. The Bit timing parameters can be configured individually for each CAN node, creating a common Bit rate even though the CAN nodes' oscillator periods ( $F_{osc}$ ) may be different.

The frequencies of these oscillators are not absolutely stable. Small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range ( $df$ ), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

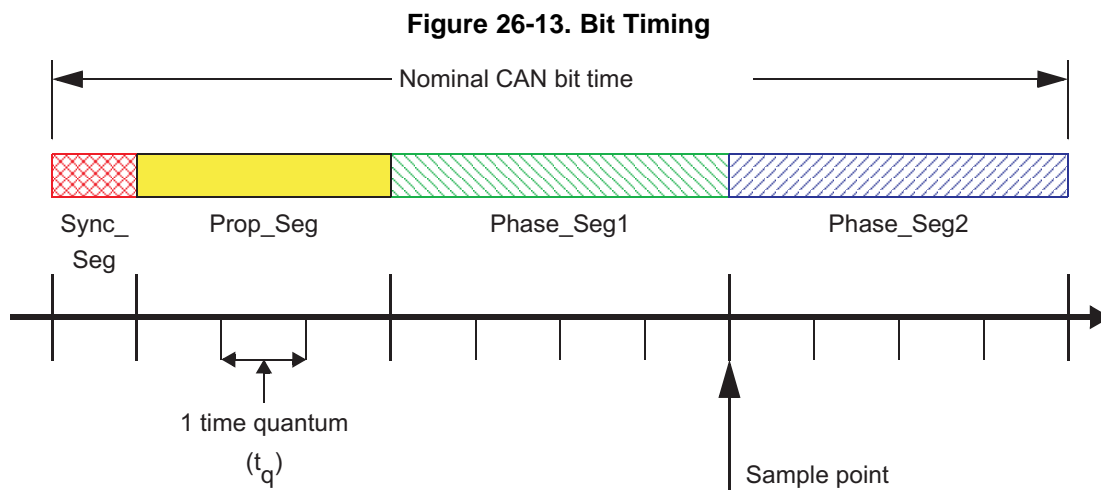
The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

### 26.13.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see Figure 26-13):

- Synchronization Segment (Sync\_Seg)
- Propagation Time Segment (Prop\_Seg)
- Phase Buffer Segment 1 (Phase\_Seg1)
- Phase Buffer Segment 2 (Phase\_Seg2)



Each segment consists of a specific number of time quanta. The length of one time quantum ( $t_q$ ), which is the basic time unit of the bit time, is given by the CAN\_CLK and the Baud Rate Prescalers (BRPE and BRP). With these two Baud Rate Prescalers combined, divider values from 1 to 1024 can be programmed:

$$t_q = \frac{\text{Baud Rate}}{\text{Prescaler} / \text{CAN\_CLK}}$$

Apart from the fixed length of the synchronization segment, these numbers are programmable. Table 26-4 describes the minimum programmable ranges required by the CAN protocol.

A given bit rate may be met by different bit time configurations.

**Table 26-4. Programmable Ranges Required by CAN Protocol**

Parameter	Range	Remark
Sync_Seg	1 $t_q$ (fixed)	Synchronization of bus input to CAN_CLK
Prop_Seg	[1 ... 8] $t_q$	Compensates for the physical delay times
Phase_Seg1	[1 ... 8] $t_q$	May be lengthened temporarily by synchronization
Phase_Seg2	[1 ... 8] $t_q$	May be shortened temporarily by synchronization
Synchronization Jump Width (SJW)	[1 ... 4] $t_q$	May not be longer than either phase buffer segment

**NOTE:** For proper functionality of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

### 26.13.1.1 Synchronization Segment

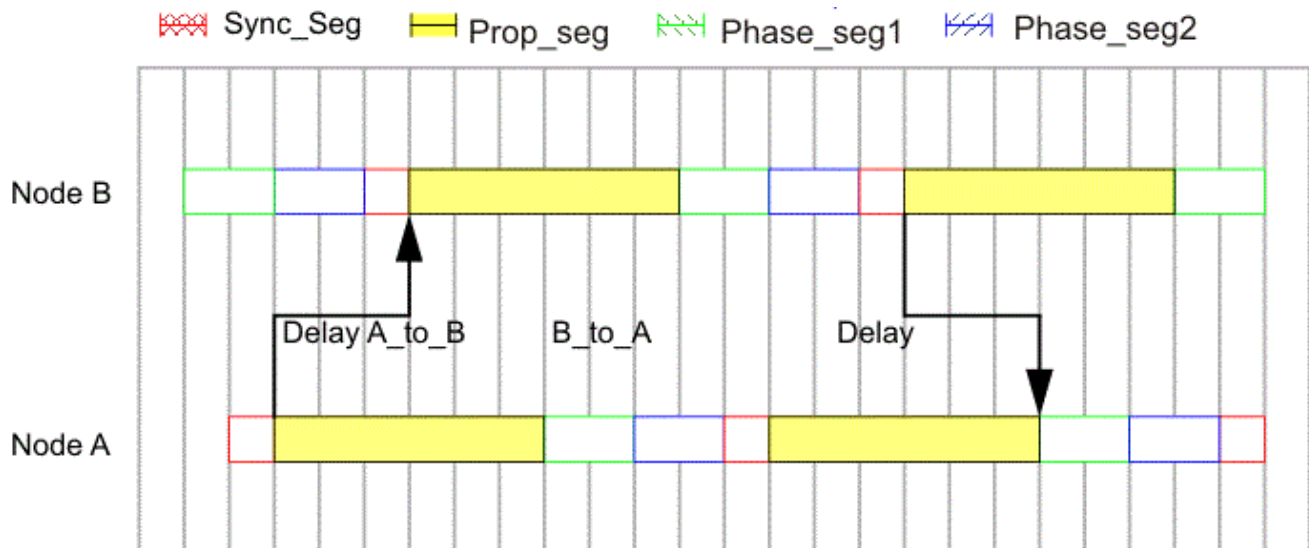
The Synchronization Segment (Sync\_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync\_Seg, its distance to the Sync\_Seg is called the phase error of this edge.

### 26.13.1.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the CAN network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus can be out of phase with the transmitter of the bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's nondestructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages require that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in [Figure 26-14](#) shows the phase shift and propagation times between two CAN nodes.

**Figure 26-14. The Propagation Time Segment**



$$\text{Delay A\_to\_B} \geq \text{node output delay(A)} + \text{bus line delay (A->B)} + \text{node input delay(B)}$$

$$\text{Prop\_Seg} \geq \text{Delay A\_to\_B} + \text{Delay B\_to\_A}$$

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. The node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay(A\_to\_B) after it has been transmitted, node B's bit timing segments are shifted with regard to node A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay(B\_to\_A).

Due to oscillator tolerances, the actual position of node A's Sample Point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase\_Seg1. This condition defines the length of Prop\_Seg.

If the edge from recessive to dominant transmitted by node B would arrive at node A after the start of Phase\_Seg1, it could happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

This error only occurs when two nodes arbitrate for the CAN bus which have oscillators of opposite ends of the tolerance range and are separated by a long bus line; this is an example of a minor error in the Bit timing configuration (Prop\_Seg too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3-Sample Mode. The CAN module on this device does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of  $1 t_q$ , requiring a longer Prop\_Seg.

### 26.13.1.3 Phase Buffer Segments and Synchronization

The phase buffer segments (Phase\_Seg1 and Phase\_Seg2) and the synchronization jump width (SJW) are used to compensate for the oscillator tolerance.

The phase buffer segments surround the sample point. The phase buffer segments may be lengthened or shortened by synchronization.

The synchronization jump width (SJW) defines how far the resynchronizing mechanism may move the sample point inside the limits defined by the phase buffer segments to compensate for edge phase errors.

Synchronizations occur on edges from recessive to dominant. Their purpose is to control the distance between edges and sample points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous sample point. A synchronization may be done only if a recessive bit was sampled at the previous sample point and if the actual time quantum's bus level is dominant.

An edge is synchronous if it occurs inside of Sync\_Seg, otherwise its distance to the Sync\_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync\_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist: hard synchronization and resynchronizing. A hard synchronization is done once at the start of a frame; inside a frame only resynchronization is possible.

- **Hard Synchronization**

After a hard synchronization, the bit time is restarted with the end of Sync\_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.

- **Bit Resynchronizations**

Resynchronization leads to a shortening or lengthening of the bit time such that the position of the sample point is shifted with regard to the edge.

When the phase error of the edge which causes resynchronization is positive, Phase\_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase\_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

When the phase error of the edge which causes resynchronization is negative, Phase\_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase\_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

If the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of hard synchronization and resynchronization are the same. If the magnitude of the phase error is larger than SJW, the resynchronization cannot compensate the phase error completely, and an error of (phase error - SJW) remains.

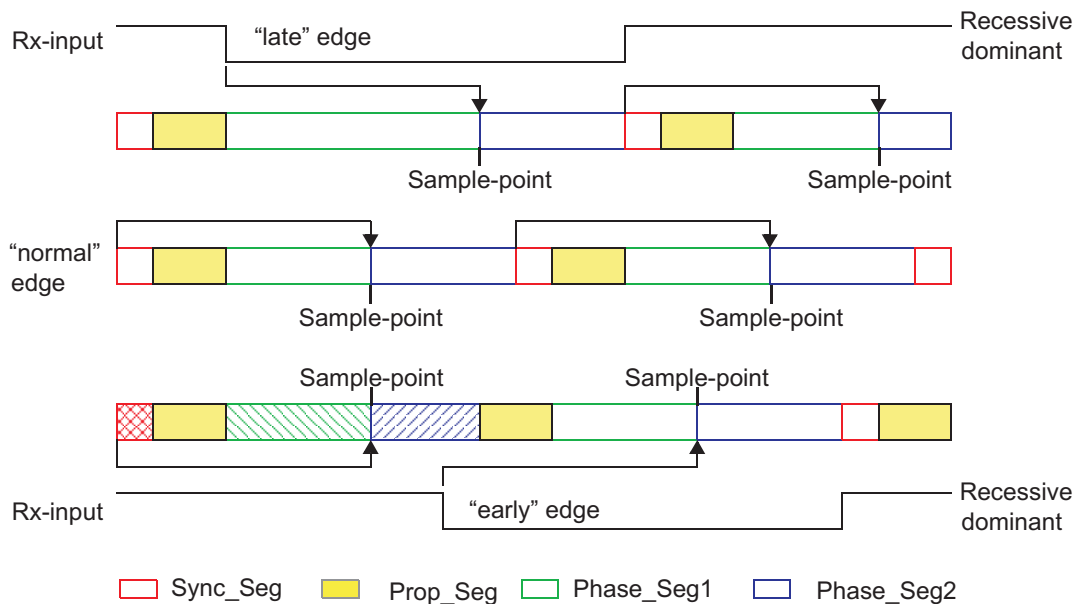
Only one synchronization may be done between two sample points. The synchronizations maintain a minimum distance between edges and sample points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop\_Seg + Phase\_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize "hard" on the edge transmitted by the "leading" transceiver that started transmitting first, but due to propagation delay times, they cannot become ideally synchronized. The "leading" transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently "take the lead" and that are differently synchronized to the previously "leading" transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that "takes the lead" in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator's clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator's tolerance range.

The examples in Figure 26-15 show how the phase buffer segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a "late" edge, the lower drawing shows the synchronization on an "early" edge, and the middle drawing is the reference without synchronization.

Figure 26-15. Synchronization on Late and Early Edges



In the first example, an edge from recessive to dominant occurs at the end of Prop\_Seg. The edge is "late" since it occurs after the Sync\_Seg. Reacting to the "late" edge, Phase\_Seg1 is lengthened so that the distance from the edge to the sample point is the same as it would have been from the Sync\_Seg to the sample point if no edge had occurred. The phase error of this "late" edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync\_Seg.

In the second example, an edge from recessive to dominant occurs during Phase\_Seg2. The edge is "early" since it occurs before a Sync\_Seg. Reacting to the "early" edge, Phase\_Seg2 is shortened and Sync\_Seg is omitted, so that the distance from the edge to the sample point is the same as it would have been from a Sync\_Seg to the sample point if no edge had occurred. As in the previous example, the magnitude of this "early" edge's phase error is less than SJW, so it is fully compensated.

The phase buffer segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

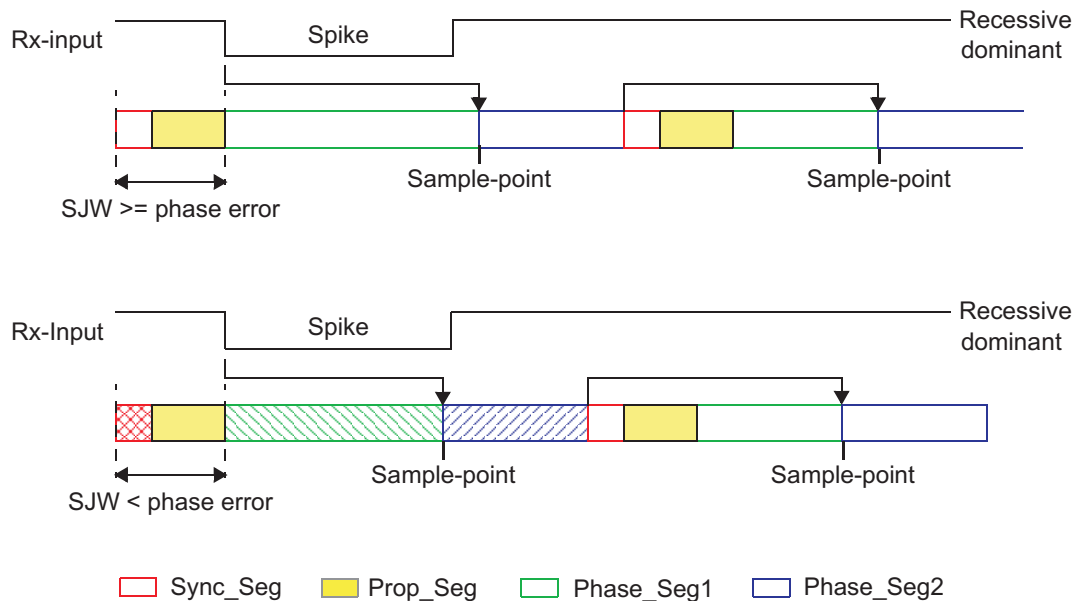
In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the sample points. The state machine omits Sync\_Seg when synchronizing on an "early" edge because it cannot subsequently redefine that time quantum of Phase\_Seg2 where the edge occurs to be the Sync\_Seg.

The examples in [Figure 26-16](#) show how short dominant noise spikes are filtered by synchronizations. In both examples, the spike starts at the end of Prop\_Seg and has the length of (Prop\_Seg + Phase\_Seg1).

In the first example, the synchronization jump width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the sample point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the sample point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

**Figure 26-16. Filtering of Short Dominant Spikes**



#### 26.13.1.4 Oscillator Tolerance Range

With the introduction of CAN protocol version 1.2, the option to synchronize on edges from dominant to recessive became obsolete. Only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range  $df$  for an oscillator's frequency  $f_{osc}$  around the nominal frequency  $f_{nom}$  with

$$(1 - df) \cdot f_{nom} \leq f_{osc} \leq (1 + df) \cdot f_{nom}$$

depends on the proportions of Phase\_Seg1, Phase\_Seg2, SJW, and the bit time. The maximum tolerance  $df$  is defined by two conditions (both shall be met):

(11)

$$df \leq \frac{\min(Tseg1, Tseg2)}{2((13 \times bit\ time) - Tseg2)}$$

$$df \leq \frac{SJW}{20 \times bit\_time}$$

(12)

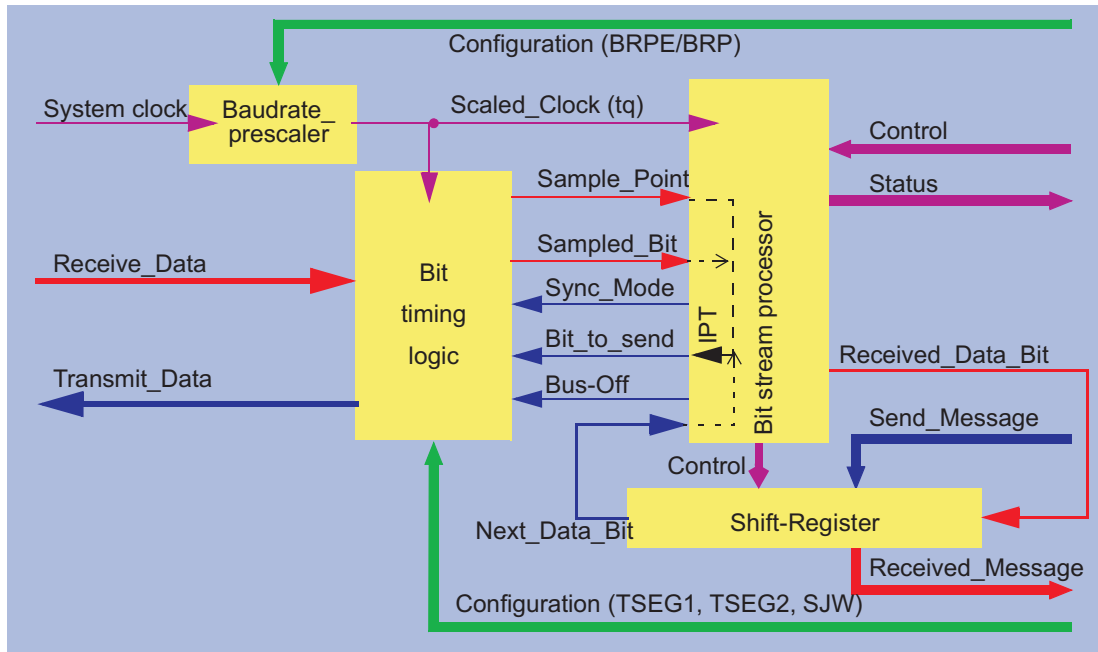
It has to be considered that SJW may not be larger than the smaller of the phase buffer segments and that the propagation time segment limits that part of the bit time that may be used for the phase buffer segments.

The combination Prop\_Seg = 1 and Phase\_Seg1 = Phase\_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8  $\mu$ s) with a bus length of 40 m.

### 26.13.2 Configuration of the CAN Bit Timing

In the CAN, the bit timing configuration is programmed in two register bytes, additionally a third byte for a baud rate prescaler extension of 4 bits (BRPE) is provided. The sum of Prop\_Seg and Phase\_Seg1 (as TSEG1) is combined with Phase\_Seg2 (as TSEG2) in one byte, SJW and BRP (plus BRPE in third byte) are combined in the other byte (see Figure 26-17).

Figure 26-17. Structure of the CAN Core's CAN Protocol Controller



In this bit timing register, the components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1...n], values in the range of [0...n-1] are programmed. That way, for example, SJW (functional range of [1...4]) is represented by only two bits.

Therefore the length of the Bit time is (programmed values)  $[TSEG1 + TSEG2 + 3] t_q$  or (functional values)  $[Sync\_Seg + Prop\_Seg + Phase\_Seg1 + Phase\_Seg2] t_q$ .

The data in the Bit Timing Register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by BRPE/BRP) defines the length of the time quantum (the basic time unit of the bit time); the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the Bit timing state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the Bit Stream Processor (BSP) state machine, is evaluated once each bit time, at the Sample Point.

The Shift register serializes the messages to be sent and parallelizes received messages. Its loading and shifting is controlled by the BSP.

The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the sample point and processes the sampled bus input bit. The time after the sample point that is needed to calculate the next bit to be sent (for example, data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT), which is  $0 t_q$  for the CAN.

Generally, the IPT is CAN controller specific, but may not be longer than  $2 t_q$ . The IPT length is the lower limit of the programmed length of Phase\_Seg2. In case of a synchronization, Phase\_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.



### 26.13.2.1 Calculation of the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting Bit time ( $1 / \text{Bit rate}$ ) must be an integer multiple of the CAN clock period.

---

**NOTE:** 8 MHz is the minimum CAN clock frequency required to operate the CAN at a bit rate of 1 MBit/s.

---

The bit time may consist of 8 to 25 time quanta. The length of the time quantum  $t_q$  is defined by the Baud Rate Prescaler with  $t_q = (\text{Baud Rate Prescaler}) / \text{CAN\_CLK}$ . Several combinations may lead to the desired bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is the Prop\_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for Prop\_Seg is converted into time quanta (rounded up to the nearest integer multiple of  $t_q$ ).

The Sync\_Seg is 1  $t_q$  long (fixed), leaving  $(\text{bit time} - \text{Prop\_Seg} - 1) t_q$  for the two Phase Buffer Segments. If the number of remaining  $t_q$  is even, the Phase Buffer Segments have the same length, Phase\_Seg2 = Phase\_Seg1, else Phase\_Seg2 = Phase\_Seg1 + 1.

The minimum nominal length of Phase\_Seg2 has to be regarded as well. Phase\_Seg2 may not be shorter than any CAN controller's Information Processing Time in the network, which is device dependent and can be in the range of  $[0 \dots 2] t_q$ .

The length of the synchronization jump width is set to its maximum value, which is the minimum of 4 and Phase\_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in [Section 26.13.1.4](#).

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration which allows the highest oscillator tolerance range.

CAN nodes with different clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol compliant configuration of the CAN bit timing.

The resulting configuration is written into the Bit Timing register:

$(\text{Phase\_Seg2}-1) \& (\text{Phase\_Seg1} + \text{Prop\_Seg} - 1) \&$

$(\text{SynchronizationJumpWidth}-1) \& (\text{Prescaler}-1)$

### 26.13.2.2 Example for Bit Timing at High Baudrate

In this example, the frequency of CAN\_CLK is 10 MHz, BRP is 0, the bit rate is 1 MBit/s.

$t_q$	100 ns	=	$t_{\text{CAN\_CLK}}$
delay of bus driver	90 ns	=	
delay of receiver circuit	40 ns	=	
delay of bus line (40m)	220 ns	=	
$t_{\text{Prop}}$	700 ns	=	$2 \cdot \text{delays} = 7 \cdot t_q$
$t_{\text{SJW}}$	100 ns	=	$1 \cdot t_q$
$t_{\text{TSeg1}}$	800 ns	=	$t_{\text{Prop}} + t_{\text{SJW}}$
$t_{\text{TSeg2}}$	100 ns	=	Information Processing Time + $1 \cdot t_q$
$t_{\text{Sync-Seg}}$	100 ns	=	$1 \cdot t_q$



bit time	1000 ns	=	$t_{\text{Sync-Seg}} + t_{\text{TSeg1}} + t_{\text{TSeg2}}$
tolerance for CAN_CLK	0.35 %	=	$\frac{\min(T\text{seg1}, T\text{seg2})}{2((13 \times \text{bit time}) - T\text{seg2})}$
			$= \frac{0.1 \mu\text{s}}{2((13 \times 1 \mu\text{s}) - 0.1 \mu\text{s})}$

In this example, the concatenated bit time parameters are  $(1-1)_3 \& (8-1)_4 \& (1-1)_2 \& (1-1)_6$ , so the Bit Timing Register is programmed to = 0x00000700.

### 26.13.2.3 Example for Bit Timing at Low Baudrate

In this example, the frequency of CAN\_CLK is 2 MHz, BRP is 1, the bit rate is 100 KBit/s.

$t_q$	1 $\mu\text{s}$	=	$2 \cdot t_{\text{CAN\_CLK}}$
delay of bus driver	200 ns	=	
delay of receiver circuit	80 ns	=	
delay of bus line (40m)	220 ns	=	
$t_{\text{Prop}}$	1 $\mu\text{s}$	=	$1 \cdot t_q$
$t_{\text{SJW}}$	4 $\mu\text{s}$	=	$4 \cdot t_q$
$t_{\text{TSeg1}}$	5 $\mu\text{s}$	=	$t_{\text{Prop}} + t_{\text{SJW}}$
$t_{\text{TSeg2}}$	4 $\mu\text{s}$	=	Information Processing Time + $4 \cdot t_q$
$t_{\text{Sync-Seg}}$	1 $\mu\text{s}$	=	$1 \cdot t_q$
bit time	10 $\mu\text{s}$	=	$t_{\text{Sync-Seg}} + t_{\text{TSeg1}} + t_{\text{TSeg2}}$
tolerance for CAN_CLK	1.58 %	=	$\frac{\min(T\text{seg1}, T\text{seg2})}{2((13 \times \text{bit time}) - T\text{seg2})}$
			$= \frac{4 \mu\text{s}}{2((13 \times 10 \mu\text{s}) - 4 \mu\text{s})}$

In this example, the concatenated bit time parameters are  $(4-1)_3 \& (5-1)_4 \& (4-1)_2 \& (2-1)_6$ , so the Bit Timing register is programmed to = 0x000034C1.

## 26.14 Message Interface Register Sets

The interface register sets control the CPU read and write accesses to the Message RAM. There are two interface register sets for read and write access (IF1 and IF2) and one Interface Register Set for read access only (IF3).

Due to the structure of the Message RAM, it is not possible to change single bits or bytes of a message object. Instead, always a complete message object in the Message RAM is accessed. Therefore the data transfer from the IF1/IF2 registers to the Message RAM requires the message handler to perform a read-modify-write cycle. First those parts of the message object that are not to be changed are read from the Message RAM into the Interface Register set, and after the update the whole content of the Interface Register set is written into the message object.

After the partial write of a message object, those parts of the Interface Register set which are not selected in the Command Register, will be set to the actual contents of the selected message object. After the partial read of a message object, those parts of the Interface Register set which are not selected in the Command Register, will be left unchanged.

By buffering the data to be transferred, the Interface Register sets avoid conflicts between concurrent CPU accesses to the Message RAM and CAN message reception and transmission. A complete message object (see [Section 26.15.1](#)) or parts of the message object may be transferred between the Message RAM and the IF1/IF2 Register set in one single transfer. This transfer, performed in parallel on all selected parts of the message object, guarantees the data consistency of the CAN message.

That being said, there is one condition that can cause a write access to the message RAM to be lost. If `MsgVal = 1` for the message object which is accessed and CAN communication is ongoing, a transfer from the IFx register to message RAM may be lost. The reason for this is that it might happen that the IFx register write to the message RAM occurs in between a read-modify-write access of the Host Message Handler when it is in the process of receiving a message for the same message object.

To avoid this issue with receive mail boxes, reset `MsgVal` before changing any of the following: `Id28-0`, `Xtd`, `Dir`, `DLC3-0`, `RxIE`, `TxIE`, `RmtEn`, `EoB`, `Umask`, `Msk28-0`, `MXtd`, and `MDir`.

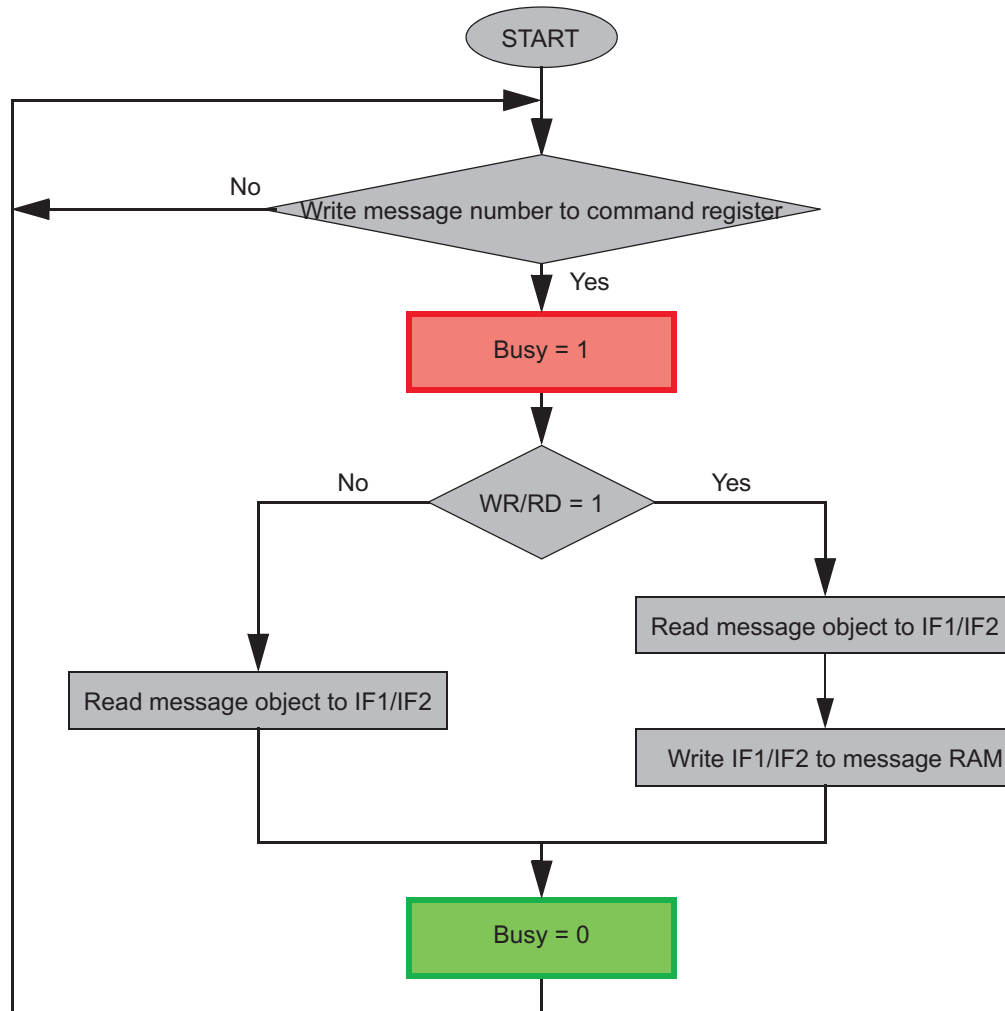
To avoid this issue with transmit mail boxes, reset `MsgVal` before changing any of the following: `Dir`, `RxIE`, `TxIE`, `RmtEn`, `EoB`, `Umask`, `Msk28-0`, `MXtd`, and `MDir`. Other fields not listed above, like `Data`, may be changed without fear of losing a write to the message RAM.

### 26.14.1 Message Interface Register Sets 1 and 2

The IF1 and IF2 register sets allow data transfers to and from the message objects. The IFxCMD register for an interface control the direction of the data transfer. If the IFxCMD register is set to write, then the message object fields selected by the IFxCMD register will be overwritten by values taken from the other IFx registers. If the IFxCMD register is set to read, then the message object fields selected by the IFxCMD register will be copied from the message object to the other IFx registers. The interfaces allow for transfers of a complete message object as well as individual parts. The transfer begins with the desired message object number is written to bits 7:0 of the IFxCMD register.

When the CPU initiates a data transfer between the IF1/IF2 registers and Message RAM, the message handler sets the Busy bit in the respective Command Register to '1'. After the transfer has completed, the Busy bit is set back to '0' (see [Figure 26-18](#) ).

Figure 26-18. Data Transfer Between IF1 / IF2 Registers and Message RAM



### 26.14.2 IF3 Register Set

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU. The automatic update functionality can be programmed for each message object (see the IF3 Update Enable register).

All valid message objects in Message RAM which are configured for automatic update, will be checked for active NewDat flags. If such a message object is found, it will be transferred to the IF3 register (if no previous DMA transfers are ongoing), controlled by IF3 Observation register. If more than one NewDat flag is active, the message object with the lowest number has the highest priority for automatic IF3 update.

The NewDat bit in the message object will be reset by a transfer to IF3.

If DCAN internal IF3 update is complete, a DMA will be requested. The DMA request stays active until the first read access to one of the IF3 registers. The DMA functionality has to be enabled by setting bit DE3 in the CAN Control register.

---

**NOTE:** The IF3 register set can not be used for transferring data into message objects.

---

## 26.15 Message RAM

The CAN Message RAM contains message objects and parity bits for the message objects. There are 32 message objects in the Message RAM.

During normal operation, accesses to the Message RAM are performed via the Interface Register sets, and the CPU cannot directly access the Message RAM.

The Interface Register sets IF1 and IF2 provide indirect read/write access from the CPU to the Message RAM. The IF1 and IF2 register sets can buffer control and user data to be transferred to and from the message objects.

The third Interface Register set IF3 can be configured to automatically receive control and user data from the Message RAM when a message object has been updated after reception of a CAN message. The CPU does not need to initiate the transfer from Message RAM to IF3 Register set.

The message handler avoids potential conflicts between concurrent accesses to Message RAM and CAN frame reception/transmission.

The message RAM can only be accessed in debug mode. The message RAM base address is 0x1000 above the base address of the CAN peripheral.

### 26.15.1 Structure of Message Objects

Figure 26-19 shows the structure of a message object.

The grayed fields are those parts of the message object which are represented in dedicated registers. For example, the transmit request flags of all message objects are represented in centralized transmit request registers.

**Figure 26-19. Structure of a Message Object**

Message Object												
UMask	Msk[28:0]	MXtd	MDir	EoB	unused	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID[28:0]	Xtd	Dir	DLC[3:0]	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7

**Table 26-5. Message Object Field Descriptions**

Name	Value	Description
MsgVal	0	Message valid The message object is ignored by the message handler.
	1	The message object is to be used by the message handler.
		Note: This bit may be kept at level '1' even when the identifier bits ID[28:0], the control bits Xtd, Dir, or the data length code DLC[3:0] are changed. It should be reset if the Messages Object is no longer required.
UMask	0	Use Acceptance Mask Mask bits (Msk[28:0], MXtd and MDir) are ignored and not used for acceptance filtering.
	1	Mask bits are used for acceptance filtering.
		Note: If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
ID[28:0]	ID[28:0]	Message Identifier 29-bit ("extended") identifier bits
	ID[28:18]	11-bit ("standard") identifier bits
Msk[28:0]	0	Identifier Mask The corresponding bit in the message identifier is not used for acceptance filtering (don't care).
	1	The corresponding bit in the message identifier is used for acceptance filtering.
		Note: The bit functionality in the DCAN module is the opposite of the Local Acceptance Mask bit functionality in the eCAN module found in older C28xx devices, where a "1" means the corresponding bit is NOT used for filtering, and "0" means it is used.

**Table 26-5. Message Object Field Descriptions (continued)**

Name	Value	Description
Xtd	0	Extended Identifier The 11-bit ("standard") identifier will be used for this message object.
	1	The 29-bit ("extended") identifier will be used for this message object.
MXtd	0	Mask Extended Identifier The extended identifier bit (IDE) has no effect on the acceptance filtering.
	1	The extended identifier bit (IDE) is used for acceptance filtering.  Note: When 11-bit ("standard") Identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
Dir	0	Message Direction Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, the message is stored in this message object.
	1	Direction = transmit: On TxRqst, a data frame is transmitted. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
MDir	0	Mask Message Direction The message direction bit (Dir) has no effect on the acceptance filtering.
	1	The message direction bit (Dir) is used for acceptance filtering.
EOB	0	End of Block The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.
	1	The message object is a single message object or the last message object in a FIFO Buffer Block.  Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.
NewDat	0	New Data No new data has been written into the data bytes of this message object by the message handler since the last time when this flag was cleared by the CPU.
	1	The message handler or the CPU has written new data into the data bytes of this message object.
MsgLst	0	Message Lost (only valid for Message Objects with direction = receive) No message was lost since the last time when this bit was reset by the CPU.
	1	The message handler stored a new message into this message object when NewDat was still set, so the previous message has been overwritten.
RxIE	0	Receive Interrupt Enable IntPnd will not be triggered after the successful reception of a frame.
	1	IntPnd will be triggered after the successful reception of a frame.
TxIE	0	Transmit Interrupt Enable IntPnd will not be triggered after the successful transmission of a frame.
	1	IntPnd will be triggered after the successful transmission of a frame.
IntPnd	0	Interrupt Pending This message object is not the source of an interrupt.
	1	This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.
RmtEn	0	Remote Enable At the reception of a remote frame, TxRqst is not changed.
	1	At the reception of a remote frame, TxRqst is set.  Note: See <a href="#">Section 26.12.8</a> for details on the setup of RmtEn and UMask for remote frames.
TxRqst	0	Transmit Request This message object is not waiting for a transmission.
	1	The transmission of this message object is requested and is not yet done.

**Table 26-5. Message Object Field Descriptions (continued)**

Name	Value	Description
DLC[3:0]	0-8 9-15	Data length code Data frame has 0-8 data bytes. Data frame has 8 data bytes.  Note: The data length code of a message object must be defined to the same value as in the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.
Data 0 Data 1 Data 2 Data 3 Data 4 Data 5 Data 6 Data 7		1st data byte of a CAN data frame 2nd data byte of a CAN data frame 3rd data byte of a CAN data frame 4th data byte of a CAN data frame 5th data byte of a CAN data frame 6th data byte of a CAN data frame 7th data byte of a CAN data frame 8th data byte of a CAN data frame  Note: Byte Data 0 is the first data byte shifted into the shift register of the CAN Core during a reception, byte Data 7 is the last. When the message handler stores a data frame, it will write all the eight data bytes into a message object. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by undefined values.

### 26.15.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:

Message RAM base address + (message object number) \* 0x20.

This means that Message Object 1 starts at offset 0x0020; Message Object 2 starts at offset 0x0040, and so on.

---

**NOTE:** '0' is not a valid message object number. At address 0x0000, the last message object (32) (with the lowest priority) is located. Writing to the address of an unimplemented message object may overwrite an implemented message object.

---

Message Object number 1 has the highest priority.

**Table 26-6. Message RAM Addressing in Debug Mode**

Message Object Number	Offset From Base Address	Word Number	Debug Mode <sup>(1)</sup>
last implemented (here:32)	0x0000	1	Parity
	0x0004	2	MXtd,MDir,Mask
	0x0008	3	Xtd,Dir,ID
	0x000C	4	Ctrl
	0x0010	5	Data Bytes 3-0
	0x0014	6	Data Bytes 7-4
1	0x0020	1	Parity
	0x0024	2	MXtd,MDir,Mask
	0x0028	3	Xtd,Dir,ID
	0x002C	4	Ctrl
	0x0030	5	Data Bytes 3-0
	0x0034	6	Data Bytes 7-4

<sup>(1)</sup> See [Section 26.15.3](#).

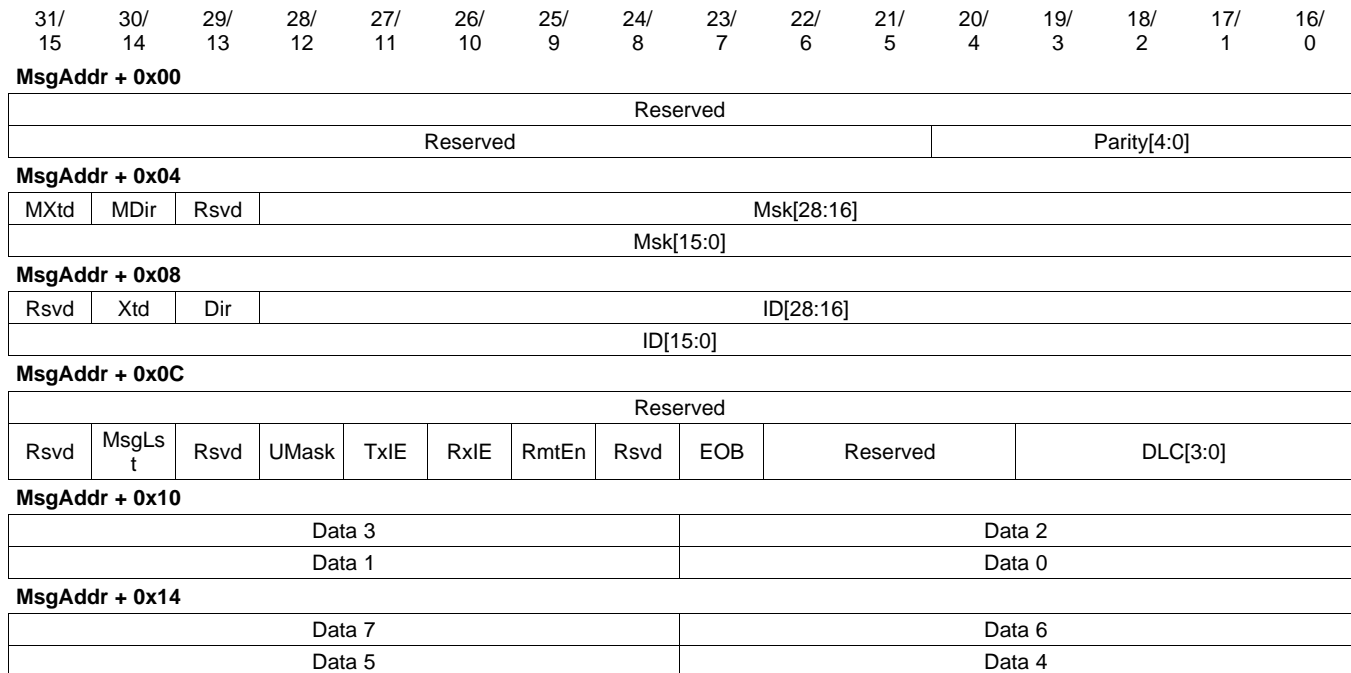
**Table 26-6. Message RAM Addressing in Debug Mode (continued)**

Message Object Number	Offset From Base Address	Word Number	Debug Mode <sup>(1)</sup>
2	0x0040	1	Parity
	0x0044	2	MXtd,MDir,Mask
	0x0048	3	Xtd,Dir,ID
	0x004C	4	Ctrl
	0x0050	5	Data Bytes 3-0
	0x0054	6	Data Bytes 7-4
...	...	...	...
31	0x03E0	1	Parity
	0x03E4	2	MXtd,MDir,Mask
	0x03E8	3	Xtd,Dir,ID
	0x03EC	4	Ctrl
	0x03F0	5	Data Bytes 3-0
	0x03F4	6	Data Bytes 7-4

### 26.15.3 Message RAM Representation in Debug Mode

In debug mode, the Message RAM will be memory mapped. This allows the external debug unit to access the Message RAM.

**NOTE:** During debug mode, the Message RAM cannot be accessed via the IFx register sets.

**Figure 26-20. Message RAM Representation in Debug Mode**

## 26.16 CAN Registers

This section describes the Controller Area Network registers.

### 26.16.1 CAN Base Addresses

**Table 26-7. CAN Base Address Table**

Device Registers	Register Name	Start Address	End Address
CanaRegs	CAN_REGS	0x0004_8000	0x0004_87FF
CanbRegs	CAN_REGS	0x0004_A000	0x0004_A7FF



## 26.16.2 CAN\_REGS Registers

Table 26-8 lists the CAN\_REGS registers. All register offset addresses not listed in Table 26-8 should be considered as reserved locations and the register contents should not be modified.

**Table 26-8. CAN\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	CAN_CTL	CAN Control Register		<a href="#">Go</a>
4h	CAN_ES	Error and Status Register		<a href="#">Go</a>
8h	CAN_ERRC	Error Counter Register		<a href="#">Go</a>
Ch	CAN_BTR	Bit Timing Register		<a href="#">Go</a>
10h	CAN_INT	Interrupt Register		<a href="#">Go</a>
14h	CAN_TEST	Test Register		<a href="#">Go</a>
1Ch	CAN_PERR	CAN Parity Error Code Register		<a href="#">Go</a>
40h	CAN_RAM_INIT	CAN RAM Initialization Register		<a href="#">Go</a>
50h	CAN_GLB_INT_EN	CAN Global Interrupt Enable Register		<a href="#">Go</a>
54h	CAN_GLB_INT_FLG	CAN Global Interrupt Flag Register		<a href="#">Go</a>
58h	CAN_GLB_INT_CLR	CAN Global Interrupt Clear Register		<a href="#">Go</a>
80h	CAN_ABOTR	Auto-Bus-On Time Register		<a href="#">Go</a>
84h	CAN_TXRQ_X	CAN Transmission Request Register		<a href="#">Go</a>
88h	CAN_TXRQ_21	CAN Transmission Request 2_1 Register		<a href="#">Go</a>
98h	CAN_NDAT_X	CAN New Data Register		<a href="#">Go</a>
9Ch	CAN_NDAT_21	CAN New Data 2_1 Register		<a href="#">Go</a>
ACh	CAN_IPEN_X	CAN Interrupt Pending Register		<a href="#">Go</a>
B0h	CAN_IPEN_21	CAN Interrupt Pending 2_1 Register		<a href="#">Go</a>
C0h	CAN_MVAL_X	CAN Message Valid Register		<a href="#">Go</a>
C4h	CAN_MVAL_21	CAN Message Valid 2_1 Register		<a href="#">Go</a>
D8h	CAN_IP_MUX21	CAN Interrupt Multiplexer 2_1 Register		<a href="#">Go</a>
100h	CAN_IF1CMD	IF1 Command Register		<a href="#">Go</a>
104h	CAN_IF1MSK	IF1 Mask Register		<a href="#">Go</a>
108h	CAN_IF1ARB	IF1 Arbitration Register		<a href="#">Go</a>
10Ch	CAN_IF1MCTL	IF1 Message Control Register		<a href="#">Go</a>
110h	CAN_IF1DATA	IF1 Data A Register		<a href="#">Go</a>
114h	CAN_IF1DATB	IF1 Data B Register		<a href="#">Go</a>
120h	CAN_IF2CMD	IF2 Command Register		<a href="#">Go</a>
124h	CAN_IF2MSK	IF2 Mask Register		<a href="#">Go</a>
128h	CAN_IF2ARB	IF2 Arbitration Register		<a href="#">Go</a>
12Ch	CAN_IF2MCTL	IF2 Message Control Register		<a href="#">Go</a>
130h	CAN_IF2DATA	IF2 Data A Register		<a href="#">Go</a>
134h	CAN_IF2DATB	IF2 Data B Register		<a href="#">Go</a>
140h	CAN_IF3OBS	IF3 Observation Register		<a href="#">Go</a>
144h	CAN_IF3MSK	IF3 Mask Register		<a href="#">Go</a>
148h	CAN_IF3ARB	IF3 Arbitration Register		<a href="#">Go</a>
14Ch	CAN_IF3MCTL	IF3 Message Control Register		<a href="#">Go</a>
150h	CAN_IF3DATA	IF3 Data A Register		<a href="#">Go</a>
154h	CAN_IF3DATB	IF3 Data B Register		<a href="#">Go</a>
160h	CAN_IF3UPD	IF3 Update Enable Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 26-9 shows the codes that are used for access types in this section.

**Table 26-9. CAN\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
W1S	W 1S	Write 1 to set
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**26.16.2.1 CAN\_CTL Register (Offset = 0h) [reset = 1401h]**

CAN\_CTL is shown in [Figure 26-21](#) and described in [Table 26-10](#).

Return to the [Summary Table](#).

This register is used for configuring the CAN module in terms of interrupts, parity, debug-mode behavior etc.

**Figure 26-21. CAN\_CTL Register**

31	30	29	28	27	26	25	24
RESERVED						RESERVED	RESERVED
R-0h							
23	22	21	20	19	18	17	16
RESERVED			DE3	DE2	DE1	IE1	INITDBG
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h
15	14	13	12	11	10	9	8
SWR	RESERVED	PMD				ABO	IDS
R/W-0h	R-0h	R/W-5h				R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
Test	CCE	DAR	RESERVED	EIE	SIE	IE0	Init
R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h

**Table 26-10. CAN\_CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23-21	RESERVED	R	0h	Reserved
20	DE3	R/W	0h	Enable DMA request line for IF3 0 Disabled 1 Enabled Note: A pending DMA request for IF3 remains active until first access to one of the IF3 registers. Reset type: SYSRSn
20	RESERVED	R/W	0h	Reserved
19	DE2	R/W	0h	Enable DMA request line for IF2 0 Disabled 1 Enabled Note: A pending DMA request for IF1 remains active until first access to one of the IF2 registers. Reset type: SYSRSn
19	RESERVED	R/W	0h	Reserved
18	DE1	R/W	0h	Enable DMA request line for IF1 0 Disabled 1 Enabled Note: A pending DMA request for IF1 remains active until first access to one of the IF1 registers. Reset type: SYSRSn
18	RESERVED	R/W	0h	Reserved

**Table 26-10. CAN\_CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	IE1	R/W	0h	Interrupt line 1 Enable  0 CANINT1 is disabled.  1 CANINT1 is enabled. Interrupts will assert CANINT1 line to 1 line remains active until pending interrupts are processed. Reset type: SYSRSn
16	INITDBG	R	0h	Debug Mode Status Bit: This bit indicates the internal init state for a debug access  0 Not in debug mode, or debug mode requested but not entered.  1 Debug mode requested and internally entered the CAN module is ready for debug accesses. Reset type: SYSRSn
15	SWR	R/W	0h	Software Reset Enable Bit: This bit activates the software reset.  0 Normal Operation.  1 Module is forced to reset state. This bit will get cleared automatically one clock cycle after execution of software reset.  Note: To execute software reset, the following procedure is necessary: 1. Set INIT bit to shut down CAN communication. 2. Set SWR bit. This bit is EALLOW protected. Note: This bit is write-protected by Init bit Reset type: SYSRSn
14	RESERVED	R	0h	Reserved
13-10	PMD	R/W	5h	Parity on/off  0101 Parity function disabled  Any other value - Parity function enabled Reset type: SYSRSn
9	ABO	R/W	0h	Auto-Bus-On Enable  0 The Auto-Bus-On feature is disabled  1 The Auto-Bus-On feature is enabled Reset type: SYSRSn
8	IDS	R/W	0h	Interruption Debug Support Enable  0 When Debug mode is requested, the CAN module will wait for a started transmission or reception to be completed before entering Debug mode  1 When Debug mode is requested, the CAN module will interrupt any transmission or reception, and enter Debug mode immediately. Reset type: SYSRSn
7	Test	R/W	0h	Test Mode Enable  0 Disable Test Mode (Normal operation)  1 Enable Test Mode Reset type: SYSRSn

**Table 26-10. CAN\_CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	CCE	R/W	0h	<p>Configuration Change Enable</p> <p>0 The CPU has no write access to the configuration registers.</p> <p>1 The CPU has write access to the configuration registers (when Init bit is set).</p> <p>Reset type: SYSRSn</p>
5	DAR	R/W	0h	<p>Disable Automatic Retransmission</p> <p>0 Automatic Retransmission of "not successful" messages enabled.</p> <p>1 Automatic Retransmission disabled.</p> <p>Reset type: SYSRSn</p>
4	RESERVED	R	0h	Reserved
3	EIE	R/W	0h	<p>Error Interrupt Enable</p> <p>0 Disabled - PER, BOff and EWarn bits cannot generate an interrupt.</p> <p>1 Enabled - PER, BOff and EWarn bits can generate an interrupt at CANINT0 line and affect the Interrupt Register.</p> <p>Reset type: SYSRSn</p>
2	SIE	R/W	0h	<p>Status Change Interrupt Enable</p> <p>0 Disabled - RxOk, TxOk and LEC bits cannot generate an interrupt.</p> <p>1 Enabled - RxOk, TxOk and LEC can generate an interrupt on the CANINT0 line</p> <p>Reset type: SYSRSn</p>
1	IE0	R/W	0h	<p>Interrupt line 0 Enable</p> <p>0 CANINT0 is disabled.</p> <p>1 CANINT0 is enabled. Interrupts will assert CANINT0 line to 1 line remains active until pending interrupts are processed.</p> <p>Reset type: SYSRSn</p>
0	Init	R/W	1h	<p>Initialization Mode</p> <p>This bit is used to keep the CAN module inactive during bit timing configuration and message RAM initialization. It is set automatically during a bus off event. Clearing this bit will not shorten the bus recovery time.</p> <p>0 CAN module processes messages normally</p> <p>1 CAN module ignores bus activity</p> <p>Reset type: SYSRSn</p>

### 26.16.2.2 CAN\_ES Register (Offset = 4h) [reset = 7h]

CAN\_ES is shown in [Figure 26-22](#) and described in [Table 26-11](#).

Return to the [Summary Table](#).

This register indicates error conditions, if any, of the CAN module. Interrupts are generated by PER, BOff and EWarn bits (if EIE bit in CAN Control Register is set) and by RxOk, TxOk, and LEC bits (if SIE bit in CAN Control Register is set). A change of bit EPass will not generate an Interrupt.

Reading the Error and Status Register clears the PER, RxOk and TxOk bits and sets the LEC to value '7'. Additionally, the Status Interrupt value (0x8000) in the Interrupt Register will be replaced by the next lower priority interrupt value.

For debug support, the auto clear functionality of Error and Status Register (clear of status flags by read) is disabled when in Debug/Suspend mode.

**Figure 26-22. CAN\_ES 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					RESERVED	RESERVED	PER
R-0h							R-0h
7	6	5	4	3	2	1	0
BOff	EWarn	EPass	RxOk	TxOk	LEC		
R-0h	R-0h	R-0h	R-0h	R-0h	R-7h		

**Table 26-11. CAN\_ES Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	PER	R	0h	Parity Error Detected: This bit will be reset after the CPU reads the register.  0 No parity error has been detected since last read access.  1 The parity check mechanism has detected a parity error in the Message RAM. Reset type: SYSRSn
7	BOff	R	0h	Bus-off Status Bit:  0 The CAN module is not in Bus-Off state.  1 The CAN module is in Bus-Off state. Reset type: SYSRSn
6	EWarn	R	0h	Warning State Bit:  0 Both error counters are below the error warning limit of 96.  1 At least one of the error counters has reached the error warning limit of 96. Reset type: SYSRSn

**Table 26-11. CAN\_ES Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	EPass	R	0h	<p>Error Passive State</p> <p>0 On CAN Bus error, the CAN could send active error frames.</p> <p>1 The CAN Core is in the error passive state as defined in the CAN Specification.</p> <p>Reset type: SYSRSn</p>
4	RxOk	R	0h	<p>Reception status Bit: This bit indicates the status of reception. The bit will be reset after the CPU reads the register.</p> <p>0 No message has been successfully received since the last time when this bit was read by the CPU. This bit is never reset by CAN internal events.</p> <p>1 A message has been successfully received since the last time when this bit was reset by a read access of the CPU. This bit will be set independent of the result of acceptance filtering.</p> <p>Reset type: SYSRSn</p>
3	TxOk	R	0h	<p>Transmission status Bit: This bit indicates the status of transmission. The bit will be reset after the CPU reads the register.</p> <p>0 No message has been successfully transmitted since the last time when this bit was read by the CPU. This bit is never reset by CAN internal events.</p> <p>1 A message has been successfully transmitted (error free and acknowledged by at least one other node) since the last time when this bit was cleared by a read access of the CPU.</p> <p>Reset type: SYSRSn</p>
2-0	LEC	R	7h	<p>Last Error Code</p> <p>The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. This field will be reset to '7' whenever the CPU reads the register.</p> <p>0 No Error</p> <p>1 Stuff Error: More than five equal bits in a row have been detected in a part of a received message where this is not allowed.</p> <p>2 Form Error: A fixed format part of a received frame has the wrong format.</p> <p>3 Ack Error: The message this CAN Core transmitted was not acknowledged by another node.</p> <p>4 Bit1 Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.</p> <p>5 Bit0 Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value '0'), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).</p> <p>6 CRC Error: In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data).</p> <p>7 No CAN bus event was detected since the last time when CPU has read the Error and Status Register. Any read access to the Error and Status Register re-initializes the LEC to value '7'.</p> <p>Reset type: SYSRSn</p>

### 26.16.2.3 CAN\_ERRC Register (Offset = 8h) [reset = 0h]

CAN\_ERRC is shown in [Figure 26-23](#) and described in [Table 26-12](#).

Return to the [Summary Table](#).

This register reflects the value of the Transmit and Receive error counters

**Figure 26-23. CAN\_ERRC 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
RP		REC						TEC							
R-0h		R-0h						R-0h							

**Table 26-12. CAN\_ERRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	RP	R	0h	Receive Error Passive  0 The Receive Error Counter is below the error passive level. 1 The Receive Error Counter has reached the error passive level as defined in the CAN Specification. Reset type: SYSRSn
14-8	REC	R	0h	Receive Error Counter  Actual state of the Receive Error Counter (values from 0 to 127). Reset type: SYSRSn
7-0	TEC	R	0h	Transmit Error Counter  Actual state of the Transmit Error Counter. (values from 0 to 255). Reset type: SYSRSn



**26.16.2.4 CAN\_BTR Register (Offset = Ch) [reset = 2301h]**

CAN\_BTR is shown in [Figure 26-24](#) and described in [Table 26-13](#).

Return to the [Summary Table](#).

This register is used to configure the bit-timing parameters for the CAN module. This register is only writable if CCE and Init bits in the CAN Control Register are set.

The CAN bit time may be programmed in the range of 8 to 25 time quanta.

The CAN time quantum may be programmed in the range of 1 to 1024 CAN\_CLK periods.

**Figure 26-24. CAN\_BTR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED				BRPE			
R-0h				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED		TSEG2			TSEG1		
R-0h		R/W-2h			R/W-3h		
7	6	5	4	3	2	1	0
SJW			BRP				
R/W-0h			R/W-1h				

**Table 26-13. CAN\_BTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-16	BRPE	R/W	0h	Baud Rate Prescaler Extension Valid programmed values are 0 to 15. By programming BRPE the Baud Rate Prescaler can be extended to values up to 1024. Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn
15	RESERVED	R	0h	Reserved
14-12	TSEG2	R/W	2h	Time segment after the sample point Valid programmed values are 0 to 7. The actual TSeg2 value which is interpreted for the Bit Timing will be the programmed TSeg2 value + 1. Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn
11-8	TSEG1	R/W	3h	Time segment before the sample point Valid programmed values are 1 to 15. The actual TSeg1 value interpreted for the Bit Timing will be the programmed TSeg1 value + 1. Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn
7-6	SJW	R/W	0h	Synchronization Jump Width Valid programmed values are 0 to 3. The actual SJW value interpreted for the Synchronization will be the programmed SJW value + 1. Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn

**Table 26-13. CAN\_BTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-0	BRP	R/W	1h	Baud Rate Prescaler-  Value by which the CAN_CLK frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta.  Valid programmed values are 0 to 63.  The actual BRP value interpreted for the Bit Timing will be the programmed BRP value + 1.  Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn

**26.16.2.5 CAN\_INT Register (Offset = 10h) [reset = 0h]**

CAN\_INT is shown in [Figure 26-25](#) and described in [Table 26-14](#).

Return to the [Summary Table](#).

This register is used to identify the source of the interrupt(s).

**Figure 26-25. CAN\_INT 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								INT1ID								INT0ID															
R-0h								R-0h								R-0h															

**Table 26-14. CAN\_INT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	INT1ID	R	0h	<p>Interrupt 1 Cause</p> <p>0x00 No interrupt is pending.</p> <p>0x01-0x20 Number of message object (mailbox) which caused the interrupt.</p> <p>0x21-0xFF Unused.</p> <p>If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority.</p> <p>Note: The CANINT1 interrupt line remains active until INT1ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared. A message interrupt is cleared by clearing the mailbox's IntPnd bit. Among the message interrupts, the mailbox's interrupt priority decreases with increasing message number.</p> <p>Reset type: SYSRSn</p>
15-0	INT0ID	R	0h	<p>Interrupt 0 Cause</p> <p>0x0000 - No interrupt is pending.</p> <p>0x0001 - 0x0020 - Number of message object which caused the interrupt.</p> <p>0x0021 - 0x7FFF - Unused.</p> <p>0x8000 - Error and Status Register value is not 0x07.</p> <p>0x8001 - 0xFFFF - Unused.</p> <p>If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority.</p> <p>Note: The CANINT0 interrupt line remains active until INT0ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared. The Status Interrupt has the highest priority. Among the message interrupts, the message object's interrupt priority decreases with increasing message number.</p> <p>Reset type: SYSRSn</p>

**26.16.2.6 CAN\_TEST Register (Offset = 14h) [reset = 0h]**

CAN\_TEST is shown in [Figure 26-26](#) and described in [Table 26-15](#).

Return to the [Summary Table](#).

This register is used to configure the various test options supported. For all test modes, the Test bit in CAN Control Register needs to be set to one. If Test bit is set, the RDA, EXL, Tx1, Tx0, LBack and Silent bits are writable. Bit Rx monitors the state of CANRX pin and therefore is only readable. All Test Register functions are disabled when Test bit is cleared.

Note: Setting Tx[1:0] other than '00' will disturb message transfer.

Note: When the internal loop back mode is active (bit LBack is set), bit EXL will be ignored.

**Figure 26-26. CAN\_TEST 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						RDA	EXL
R-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RX	TX		LBACK	SILENT	RESERVED		
R-0h	R/W-0h		R/W-0h	R/W-0h	R-0h		

**Table 26-15. CAN\_TEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	RDA	R/W	0h	RAM Direct Access Enable: 0 Normal Operation. 1 Direct access to the RAM is enabled while in Test Mode. Reset type: SYSRSn
8	EXL	R/W	0h	External Loop Back Mode: 0 Disabled. 1 Enabled. Reset type: SYSRSn
7	RX	R	0h	Monitors the actual value of the CANRX pin: 0 The CAN bus is dominant. 1 The CAN bus is recessive. Reset type: SYSRSn
6-5	TX	R/W	0h	Control of CANTX pin: 00 Normal operation, CANTX is controlled by the CAN Core. 01 Sample Point can be monitored at CANTX pin. 10 CANTX pin drives a dominant value. 11 CANTX pin drives a recessive value. Reset type: SYSRSn

**Table 26-15. CAN\_TEST Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	LBACK	R/W	0h	Loop Back Mode: 0 Disabled. 1 Enabled. Reset type: SYSRSn
3	SILENT	R/W	0h	Silent Mode: 0 Disabled. 1 Enabled. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

### 26.16.2.7 CAN\_PERR Register (Offset = 1Ch) [reset = 100h]

CAN\_PERR is shown in [Figure 26-27](#) and described in [Table 26-16](#).

Return to the [Summary Table](#).

This register indicates the Word/Mailbox number where a parity error has been detected. If a parity error is detected, the PER flag will be set in the Error and Status Register. This bit is not reset by the parity check mechanism

it must be reset by reading the Error and Status Register. In addition to the PER flag, the Parity Error Code Register will indicate the memory area where the parity error has been detected. If more than one word with a parity error was detected, the highest word number with a parity error will be displayed. After a parity error has been detected, the register will hold the last error code until power is removed.

**Figure 26-27. CAN\_PERR 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					WORD_NUM					MSG_NUM					
R-0h					R-1h					R-0h					

**Table 26-16. CAN\_PERR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Reserved
10-8	WORD_NUM	R	1h	0x01-0x05 Word number where parity error has been detected. RDA word number (1 to 5) of the mailbox (according to the Message RAM representation in RDA mode). Reset type: SYSRSn
7-0	MSG_NUM	R	0h	0x01-0x21 Mailbox number where parity error has been detected Reset type: SYSRSn

**26.16.2.8 CAN\_RAM\_INIT Register (Offset = 40h) [reset = 5h]**

CAN\_RAM\_INIT is shown in [Figure 26-28](#) and described in [Table 26-17](#).

Return to the [Summary Table](#).

This register is used to initialize the Mailbox RAM. It clears the entire mailbox RAM, including the MsgVal bits.

**Figure 26-28. CAN\_RAM\_INIT 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		RAM_INIT_DONE	CAN_RAM_INIT	KEY3	KEY2	KEY1	KEY0
R-0h		R-0h	R/W-0h	R/W-0h	R/W-1h	R/W-0h	R/W-1h

**Table 26-17. CAN\_RAM\_INIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	RAM_INIT_DONE	R	0h	CAN Mailbox RAM initialization status: 0 Read: Initialization is on-going or initialization not initiated. 1 Read: Initialization complete Reset type: SYSRSn
4	CAN_RAM_INIT	R/W	0h	Initiate CAN Mailbox RAM initialization: 0 Read: Initialization complete or initialization not initiated. Write: No action 1 Read: Initialization is on-going Write: Initiate CAN Mailbox RAM initialization. After initialization, this bit will be automatically cleared to 0. Reset type: SYSRSn
3	KEY3	R/W	0h	See Key 0 Reset type: SYSRSn
2	KEY2	R/W	1h	See Key 0 Reset type: SYSRSn
1	KEY1	R/W	0h	See Key 0 Reset type: SYSRSn
0	KEY0	R/W	1h	KEY3-KEY0 should be 1010 for any write to this register to be valid. These bits will be restored to their reset state after the CAN RAM initialization is complete. Reset type: SYSRSn

**26.16.2.9 CAN\_GLB\_INT\_EN Register (Offset = 50h) [reset = 0h]**

CAN\_GLB\_INT\_EN is shown in [Figure 26-29](#) and described in [Table 26-18](#).

Return to the [Summary Table](#).

This register is used to enable the interrupt lines to the PIE.

**Figure 26-29. CAN\_GLB\_INT\_EN 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						GLBINT1_EN	GLBINT0_EN
R-0h						R/W-0h	R/W-0h

**Table 26-18. CAN\_GLB\_INT\_EN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	GLBINT1_EN	R/W	0h	Global Interrupt Enable for CANINT1 0 CANINT1 does not generate interrupt to PIE 1 CANINT1 generates interrupt to PIE if interrupt condition occurs Reset type: SYSRSn
0	GLBINT0_EN	R/W	0h	Global Interrupt Enable for CANINT0 0 CANINT0 does not generate interrupt to PIE 1 CANINT0 generates interrupt to PIE if interrupt condition occurs Reset type: SYSRSn



**26.16.2.10 CAN\_GLB\_INT\_FLG Register (Offset = 54h) [reset = 0h]**

CAN\_GLB\_INT\_FLG is shown in [Figure 26-30](#) and described in [Table 26-19](#).

Return to the [Summary Table](#).

This register indicates if and when the interrupt line to the PIE is active.

**Figure 26-30. CAN\_GLB\_INT\_FLG 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						INT1_FLG	INT0_FLG
R-0h						R-0h	R-0h

**Table 26-19. CAN\_GLB\_INT\_FLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	INT1_FLG	R	0h	CANINT1 Flag 0 No interrupt generated 1 Interrupt is generated due to CANINT1 (refer to CAN Interrupt Status Register for the condition) Reset type: SYSRSn
0	INT0_FLG	R	0h	CANINT0 Flag 0 No interrupt generated 1 Interrupt is generated due to CANINT0 (refer to CAN Interrupt Status Register for the condition) Reset type: SYSRSn

**26.16.2.11 CAN\_GLB\_INT\_CLR Register (Offset = 58h) [reset = 0h]**

CAN\_GLB\_INT\_CLR is shown in [Figure 26-31](#) and described in [Table 26-20](#).

Return to the [Summary Table](#).

This register is used to clear the interrupt to the PIE.

**Figure 26-31. CAN\_GLB\_INT\_CLR 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						INT1_FLG_CLR	INT0_FLG_CLR
R-0h						W-0h	W-0h

**Table 26-20. CAN\_GLB\_INT\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	INT1_FLG_CLR	W	0h	Global Interrupt flag clear for CANINT1  0 No effect  1 Write 1 to clear the corresponding bit of the Global Interrupt Flag Register and allow the PIE to receive another interrupt from CANINT1. Reset type: SYSRSn
0	INT0_FLG_CLR	W	0h	Global Interrupt flag clear for CANINT0  0 No effect  1 Write 1 to clear the corresponding bit of the Global Interrupt Flag Register and allow the PIE to receive another interrupt from CANINT0. Reset type: SYSRSn

**26.16.2.12 CAN\_ABOTR Register (Offset = 80h) [reset = 0h]**

CAN\_ABOTR is shown in [Figure 26-32](#) and described in [Table 26-21](#).

Return to the [Summary Table](#).

This register is used to introduce a variable delay before the Bus-off recovery sequence is started.

**Figure 26-32. CAN\_ABOTR 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
ABO_Time																															
R/W-0h																															

**Table 26-21. CAN\_ABOTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ABO_Time	R/W	0h	<p>Auto-Bus-On Timer</p> <p>Number of clock cycles before a Bus-Off recovery sequence is started by clearing the Init bit. "Clock" refers to the input clock to the CAN module. This function has to be enabled by setting bit ABO in CAN Control Register.</p> <p>The Auto-Bus-On timer is realized by a 32-bit counter which starts to count down to zero when the module goes Bus-Off. The counter will be reloaded with the preload value of the ABO Time register after this phase.</p> <p>NOTE: On write access to the CAN Control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted.</p> <p>NOTE: During Debug mode, running Auto-Bus-On timer will be paused.</p> <p>Reset type: SYSRSn</p>

**26.16.2.13 CAN\_TXRQ\_X Register (Offset = 84h) [reset = 0h]**

CAN\_TXRQ\_X is shown in [Figure 26-33](#) and described in [Table 26-22](#).

Return to the [Summary Table](#).

With these bits, the CPU can detect if one or more bits in the CAN Transmission Request 21 Register (CAN\_TXRQ\_21) is set. Each bit in this register represents a group of eight mailboxes. If at least one of the TxRqst bits of these message objects is set, the corresponding bit in this register will be set.

**Figure 26-33. CAN\_TXRQ\_X 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				TxRqstReg2		TxRqstReg1	
R-0h				R-0h		R-0h	

**Table 26-22. CAN\_TXRQ\_X Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-2	TxRqstReg2	R	0h	Transmit Request Register 2 flag:  Bit 2 represents byte 2 of CAN_TXRQ_21. If one or more bits in that byte are set, then bit 2 will be set.  Bit 3 represents byte 3 of CAN_TXRQ_21 Register. If one or more bits in that byte are set, then bit 3 will be set. Reset type: SYSRSn
1-0	TxRqstReg1	R	0h	Transmit Request Register 1 flag:  Bit 0 represents byte 0 of CAN_TXRQ_21 Register. If one or more bits in that byte are set, then bit 0 will be set.  Bit 1 represents byte 1 of CAN_TXRQ_21 Register. If one or more bits in that byte are set, then bit 1 will be set. Reset type: SYSRSn

**26.16.2.14 CAN\_TXRQ\_21 Register (Offset = 88h) [reset = 0h]**

CAN\_TXRQ\_21 is shown in [Figure 26-34](#) and described in [Table 26-23](#).

Return to the [Summary Table](#).

This register holds the TxRqst bits of the mailboxes. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific mailbox can be set/reset by the CPU via the IF1/IF2 message interface registers, or by the message handler after reception of a remote frame or after a successful transmission.

**Figure 26-34. CAN\_TXRQ\_21 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
TxRqst																															
R-0h																															

**Table 26-23. CAN\_TXRQ\_21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	TxRqst	R	0h	Transmission Request Bits (for all message objects)  0 No transmission has been requested for this message object.  1 The transmission of this message object is requested and is not yet done.  Note: Bit 0 is for mailbox 1, Bit 1 is for mailbox 2, Bit 2 is for mailbox 3, ..., Bit 31 is for mailbox 32  Reset type: SYSRSn

**26.16.2.15 CAN\_NDAT\_X Register (Offset = 98h) [reset = 0h]**

CAN\_NDAT\_X is shown in [Figure 26-35](#) and described in [Table 26-24](#).

Return to the [Summary Table](#).

With these bits, the CPU can detect if one or more bits in the CAN New Data 21 Register (CAN\_NDAT\_21) is set. Each bit in this register represents a group of eight mailboxes. If at least one of the NewDat bits of these mailboxes are set, the corresponding bit in this register will be set.

**Figure 26-35. CAN\_NDAT\_X 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				NewDatReg2		NewDatReg1	
R-0h				R-0h		R-0h	

**Table 26-24. CAN\_NDAT\_X Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-2	NewDatReg2	R	0h	New Data Register 2 flag:  Bit 2 represents byte 2 of CAN_NDAT_21 Register. If one or more bits in that byte are set, then bit 2 will be set.  Bit 3 represents byte 3 of CAN_NDAT_21 Register. If one or more bits in that byte are set, then bit 3 will be set.  Reset type: SYSRSn
1-0	NewDatReg1	R	0h	New Data Register 1 flag:  Bit 0 represents byte 0 of CAN_NDAT_21 Register. If one or more bits in that byte are set, then bit 0 will be set.  Bit 1 represents byte 1 of CAN_NDAT_21 Register. If one or more bits in that byte are set, then bit 1 will be set.  Reset type: SYSRSn

**26.16.2.16 CAN\_NDAT\_21 Register (Offset = 9Ch) [reset = 0h]**

CAN\_NDAT\_21 is shown in [Figure 26-36](#) and described in [Table 26-25](#).

Return to the [Summary Table](#).

This register holds the NewDat bits of all mailboxes. By reading out the NewDat bits, the CPU can check for which mailboxes the data portion was updated. The NewDat bit of a specific mailbox can be set/reset by the CPU via the IFx "Message Interface" Registers or by the Message Handler after reception of a Data Frame or after a successful transmission.

**Figure 26-36. CAN\_NDAT\_21 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
NewDat																															
R-0h																															

**Table 26-25. CAN\_NDAT\_21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	NewDat	R	0h	<p>New Data Bits (for all message objects)</p> <p>0 No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU.</p> <p>1 The message handler or the CPU has written new data into the data portion of this message object.</p> <p>Note: Bit 0 is for mailbox 1, Bit 1 is for mailbox 2, Bit 2 is for mailbox 3,...., Bit 31 is for mailbox 32</p> <p>Reset type: SYSRSn</p>

**26.16.2.17 CAN\_IPEN\_X Register (Offset = ACh) [reset = 0h]**

CAN\_IPEN\_X is shown in [Figure 26-37](#) and described in [Table 26-26](#).

Return to the [Summary Table](#).

With these bits, the CPU can detect if one or more bits in the CAN Interrupt Pending 21 Register (CAN\_IPEN\_21) is set. Each bit in this register represents a group of eight mailboxes. If at least one of the IntPnd bits of these mailboxes are set, the corresponding bit in this register will be set.

**Figure 26-37. CAN\_IPEN\_X 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				IntPndReg2		IntPndReg1	
R-0h				R-0h		R-0h	

**Table 26-26. CAN\_IPEN\_X Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-2	IntPndReg2	R	0h	Interrupt Pending Register 2 flag:  Bit 2 represents byte 2 of CAN_IPEN_21 Register. If one or more bits in that byte are set, then bit 2 will be set.  Bit 3 represents byte 3 of CAN_IPEN_21 Register. If one or more bits in that byte are set, then bit 3 will be set.  Reset type: SYSRSn
1-0	IntPndReg1	R	0h	Interrupt Pending Register 1 flag:  Bit 0 represents byte 0 of CAN_IPEN_21 Register. If one or more bits in that byte are set, then bit 0 will be set.  Bit 1 represents byte 1 of CAN_IPEN_21 Register. If one or more bits in that byte are set, then bit 1 will be set.  Reset type: SYSRSn



**26.16.2.18 CAN\_IPEN\_21 Register (Offset = B0h) [reset = 0h]**

CAN\_IPEN\_21 is shown in [Figure 26-38](#) and described in [Table 26-27](#).

Return to the [Summary Table](#).

This register holds the IntPnd bits of the mailboxes. By reading out these bits, the CPU can check for pending interrupts in the mailboxes. The IntPnd bit of a specific mailbox can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

**Figure 26-38. CAN\_IPEN\_21 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
IntPnd																															
R-0h																															

**Table 26-27. CAN\_IPEN\_21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	IntPnd	R	0h	<p>Interrupt Pending bits: This register contains the bits that indicate the pending interrupts in each one of the 32 mailboxes.</p> <p>0 This mailbox is not the source of an interrupt.</p> <p>1 This mailbox is the source of an interrupt.</p> <p>Note: Bit 0 is for mailbox 1, Bit 1 is for mailbox 2, Bit 2 is for mailbox 3,...., Bit 31 is for mailbox 32</p> <p>Reset type: SYSRSn</p>

**26.16.2.19 CAN\_MVAL\_X Register (Offset = C0h) [reset = 0h]**

CAN\_MVAL\_X is shown in [Figure 26-39](#) and described in [Table 26-28](#).

Return to the [Summary Table](#).

With these bits, the CPU can detect if one or more bits in the CAN Message Valid 2\_1 Register (CAN\_MVAL\_21) is set. Each bit in this register represents a group of eight mailboxes. If at least one of the MsgVal bits of these mailboxes are set, the corresponding bit in this register will be set.

**Figure 26-39. CAN\_MVAL\_X 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				MsgValReg2		MsgValReg1	
R-0h				R-0h		R-0h	

**Table 26-28. CAN\_MVAL\_X Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-2	MsgValReg2	R	0h	Message Valid Register 2 flag:  Bit 2 represents byte 2 of CAN_MVAL_21 Register. If one or more bits in that byte are set, then bit 2 will be set.  Bit 3 represents byte 3 of CAN_MVAL_21 Register. If one or more bits in that byte are set, then bit 3 will be set.  Reset type: SYSRSn
1-0	MsgValReg1	R	0h	Message Valid Register 1 flag:  Bit 0 represents byte 0 of CAN_MVAL_21 Register. If one or more bits in that byte are set, then bit 0 will be set.  Bit 1 represents byte 1 of CAN_MVAL_21 Register. If one or more bits in that byte are set, then bit 1 will be set.  Reset type: SYSRSn

**26.16.2.20 CAN\_MVAL\_21 Register (Offset = C4h) [reset = 0h]**

CAN\_MVAL\_21 is shown in [Figure 26-40](#) and described in [Table 26-29](#).

Return to the [Summary Table](#).

This registers hold the MsgVal bits of all mailboxes. By reading out the MsgVal bits, the CPU can check which mailbox is valid. The MsgVal bit of a specific mailbox can be set/reset by the CPU via the IF1/2 "Message Interface" Registers.

**Figure 26-40. CAN\_MVAL\_21 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
MsgValReg																															
R-0h																															

**Table 26-29. CAN\_MVAL\_21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	MsgValReg	R	0h	<p>Message Valid Bits (for all message objects)</p> <p>0 This message object is ignored by the message handler.</p> <p>1 This message object is configured and will be considered by the message handler.</p> <p>Note: Bit 0 is for mailbox 1, Bit 1 is for mailbox 2, Bit 2 is for mailbox 3,..., Bit 31 is for mailbox 32</p> <p>Reset type: SYSRSn</p>

**26.16.2.21 CAN\_IP\_MUX21 Register (Offset = D8h) [reset = 0h]**

CAN\_IP\_MUX21 is shown in [Figure 26-41](#) and described in [Table 26-30](#).

Return to the [Summary Table](#).

The IntMux bit determines for each mailbox, which of the two interrupt lines (CANINT0 or CANINT1) will be asserted when the IntPnd bit of that mailbox is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN Control Register. This will also affect the INT0ID or INT1ID flags in the Interrupt Register.

**Figure 26-41. CAN\_IP\_MUX21 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
																	IntMux														
																	R/W-0h														

**Table 26-30. CAN\_IP\_MUX21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	IntMux	R/W	0h	Interrupt Mux bits:  0 CANINT0 line is active if corresponding IntPnd flag is one. 1 CANINT1 line is active if corresponding IntPnd flag is one.  Note: Bit 0 is for mailbox 32, Bit 1 is for mailbox 1, Bit 2 is for mailbox 2,..., Bit 31 is for mailbox 31 Reset type: SYSRSn

**26.16.2.22 CAN\_IF1CMD Register (Offset = 100h) [reset = 1h]**

CAN\_IF1CMD is shown in [Figure 26-42](#) and described in [Table 26-31](#).

Return to the [Summary Table](#).

The IF1/IF2 Command Registers configure and initiate the transfer between the IF1/IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred. A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 Command Register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress. After 4 to 14 clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage. If the CPU writes to both IF1/IF2 Command Registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed.

Note: While Busy bit is one, IF1/IF2 Register sets are write protected.

Note: For debug support, the auto clear functionality of the IF1/IF2 Command Registers (clear of DMAActive flag by R/W, for devices with DMA support) is disabled during Debug/Suspend mode.

Note: If an invalid Message Number is written to bits [7:0] of the IF1/IF2 Command Register, the Message Handler may access an implemented (valid) message object instead.

**Figure 26-42. CAN\_IF1CMD Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
DIR	Mask	Arb	Control	ClrIntPnd	TXRQST	DATA_A	DATA_B
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
Busy	DMAActive	RESERVED					
R-0h	R/W-0h	R-0h					
7	6	5	4	3	2	1	0
MSG_NUM							
R/W-1h							

**Table 26-31. CAN\_IF1CMD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23	DIR	R/W	0h	Write/Read  0 Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set. That is, transfer data from the mailbox into the selected IF1/IF2 Message Buffer Registers.  1 Direction = Write: Transfer direction is from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]). That is, transfer data from the selected IF1/IF2 Message Buffer Registers to the mailbox.  The other bits of IF1/IF2 Command Mask Register have different functions depending on the transfer direction.  Note: This bit is write protected by Busy bit.  Reset type: SYSRSn

**Table 26-31. CAN\_IF1CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	Mask	R/W	0h	Access Mask Bits  0 Mask bits will not be changed  1 (Direction = Read): The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set.  1 (Direction = Write): The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
21	Arb	R/W	0h	Access Arbitration Bits  0 Arbitration bits will not be changed  1 (Direction = Read): The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set.  1 (Direction = Write): The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
20	Control	R/W	0h	Access control bits.  If the TxRqst/NewDat bit in this register(Bit [18]) is set, the TxRqst/NewDat bit in the IF1 message control register will be ignored.  0 Control bits will not be changed.  1 (Direction = Read): The message control bits will be transferred from the message object addressed by message number (Bits [7:0]) to the IF1 register set.  1 (Direction = Write): The message control bits will be transferred from the IF1 register set to the message object addressed by message number (Bits [7:0]).  Note: This bit is write protected by the Busy bit. Reset type: SYSRSn
19	ClrIntPnd	R/W	0h	Clear Interrupt Pending Bit  0 IntPnd bit will not be changed  1 (Direction = Read): Clears IntPnd bit in the message object.  1 (Direction = Write): This bit is ignored.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**Table 26-31. CAN\_IF1CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	TXRQST	R/W	0h	<p>Access Transmission Request (TxRqst) / New Data (NewDat) Bit</p> <p>0 (Direction = Read): NewDat bit will not be changed.</p> <p>0 (Direction = Write): TxRqst/NewDat bit will be handled according to the Control bit.</p> <p>1 (Direction = Read): Clears NewDat bit in the message object.</p> <p>1 (Direction = Write): Sets TxRqst/NewDat in message object.</p> <p>Note: If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to one independent of the values in IF1/IF2 Message Control Register.</p> <p>Note: A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1/IF2 Message Control Register always reflect the status before resetting them.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
17	DATA_A	R/W	0h	<p>Access Data Bytes 0-3</p> <p>0 Data Bytes 0-3 will not be changed.</p> <p>1 (Direction = Read): The Data Bytes 0-3 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Data Bytes 0-3 will be transferred from the IF1/IF2 Register set to the message object addressed by the Message Number (Bits [7:0]).</p> <p>Note: The duration of the message transfer is independent of the number of bytes to be transferred.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
16	DATA_B	R/W	0h	<p>Access Data Bytes 4-7</p> <p>0 Data Bytes 4-7 will not be changed.</p> <p>1 (Direction = Read): The Data Bytes 4-7 will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Data Bytes 4-7 will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).</p> <p>Note: The duration of the message transfer is independent of the number of bytes to be transferred.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
15	Busy	R	0h	<p>Busy Flag</p> <p>0 No transfer between IF1/IF2 Register Set and Message RAM is in progress.</p> <p>1 Transfer between IF1/IF2 Register Set and Message RAM is in progress.</p> <p>This bit is set to one after the message number has been written to bits [7:0]. IF1/IF2 Register Set will be write protected. The bit is cleared after read/write action has been finished.</p> <p>Reset type: SYSRSn</p>

**Table 26-31. CAN\_IF1CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
14	DMAActive	R/W	0h	DMA trigger status due to IF1 update.  0 No IF1 DMA request is active.  1 DMA is requested after a completed transfer between IF1 and the message RAM. The DMA request remains active until the first read or write to one of the IF1 registers  an exception is a write to Message Number (Bits [7:0]) when DMAActive is one.  Note: Due to the auto reset feature of the DMAActive bit, this bit has to be set for each subsequent DMA cycle separately.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
14	RESERVED	R/W	0h	Reserved
13-8	RESERVED	R	0h	Reserved
7-0	MSG_NUM	R/W	1h	Number of message object in Message RAM which is used for data transfer  0x00 Invalid message number  0x01-0x20 Valid message numbers  0x21-0xFF Invalid message numbers  Note: This bit is write protected by Busy bit. Reset type: SYSRSn



**26.16.2.23 CAN\_IF1MSK Register (Offset = 104h) [reset = FFFFFFFFh]**

CAN\_IF1MSK is shown in [Figure 26-43](#) and described in [Table 26-32](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Mask Registers mirror the mask bits of a message object.

Note: While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write-protected.

**Figure 26-43. CAN\_IF1MSK Register**

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

**Table 26-32. CAN\_IF1MSK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MXtd	R/W	1h	Mask Extended Identifier  0 The extended identifier bit (Xtd) has no effect on the acceptance filtering.  1 The extended identifier bit (Xtd) is used for acceptance filtering.  When 11-bit ("standard") identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
30	MDir	R/W	1h	Mask Message Direction  0 The message direction bit (Dir) has no effect on the acceptance filtering.  1 The message direction bit (Dir) is used for acceptance filtering.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
29	RESERVED	R	1h	Reserved
28-0	Msk	R/W	1FFFFFFFh	Identifier Mask-  0 The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).  1 The corresponding bit in the identifier of the message object is used for acceptance filtering.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**26.16.2.24 CAN\_IF1ARB Register (Offset = 108h) [reset = 0h]**

CAN\_IF1ARB is shown in [Figure 26-44](#) and described in [Table 26-33](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Arbitration Registers mirror the arbitration bits of a message object. The Arbitration bits ID[28:0], Xtd, and Dir are used to define the identifier and type of outgoing messages and (together with the Mask bits Msk[28:0], MXtd, and MDir) for acceptance filtering of incoming messages.

A received message is stored into the valid message object with matching identifier and Direction = receive (Data Frame) or Direction = transmit (Remote Frame).

Extended frames can be stored only in message objects with Xtd = one, standard frames in message objects with Xtd = zero.

If a received message (Data Frame or Remote Frame) matches more than one valid message objects, it is stored into the one with the lowest message number.

Note: While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write-protected.

**Figure 26-44. CAN\_IF1ARB Register**

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

**Table 26-33. CAN\_IF1ARB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MsgVal	R/W	0h	Message Valid  0 The mailbox is disabled. (The message object is ignored by the message handler).  1 The mailbox is enabled. (The message object is to be used by the message handler).  The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets the Init bit in the CAN Control Register. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
30	Xtd	R/W	0h	Extended Identifier  0 The 11-bit ("standard") Identifier is used for this message object.  1 The 29-bit ("extended") Identifier is used for this message object.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**Table 26-33. CAN\_IF1ARB Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
29	Dir	R/W	0h	Message Direction  0 Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, that frame is stored in this message object.  1 Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
28-0	ID	R/W	0h	Message Identifier  ID[28:0] 29-bit Identifier ("Extended Frame")  ID[28:18] 11-bit Identifier ("Standard Frame")  Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**26.16.2.25 CAN\_IF1MCTL Register (Offset = 10Ch) [reset = 0h]**

CAN\_IF1MCTL is shown in [Figure 26-45](#) and described in [Table 26-34](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Message Control Registers mirror the message control bits of a message object. This register has control/status bits pertaining to interrupts, acceptance mask, remote frames and FIFO option.

**Figure 26-45. CAN\_IF1MCTL 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
NewDat	MsgLst	IntPnd	UMask	TxE	RxE	RmtEn	TxRqst
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
EoB	RESERVED			DLC			
R/W-0h	R-0h			R/W-0h			

**Table 26-34. CAN\_IF1MCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	NewDat	R/W	0h	New Data  0 No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU.  1 The message handler or the CPU has written new data into the data portion of this message object.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
14	MsgLst	R/W	0h	Message Lost (only valid for message objects with direction = receive)  0 No message lost since the last time when this bit was reset by the CPU.  1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
13	IntPnd	R/W	0h	Interrupt Pending  0 This message object is not the source of an interrupt.  1 This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**Table 26-34. CAN\_IF1MCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	UMask	R/W	0h	<p>Use Acceptance Mask</p> <p>0 Mask ignored</p> <p>1 Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering</p> <p>If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
11	TxIE	R/W	0h	<p>Transmit Interrupt Enable</p> <p>0 IntPnd will not be triggered after the successful transmission of a frame.</p> <p>1 IntPnd will be triggered after the successful transmission of a frame.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
10	RxIE	R/W	0h	<p>Receive Interrupt Enable</p> <p>0 IntPnd will not be triggered after the successful reception of a frame.</p> <p>1 IntPnd will be triggered after the successful reception of a frame.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
9	RmtEn	R/W	0h	<p>Remote Enable</p> <p>0 At the reception of a remote frame, TxRqst is not changed.</p> <p>1 At the reception of a remote frame, TxRqst is set.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
8	TxRqst	R/W	0h	<p>Transmit Request</p> <p>0 This message object is not waiting for a transmission.</p> <p>1 The transmission of this message object is requested and is not yet done.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
7	EoB	R/W	0h	<p>End of Block</p> <p>0 The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.</p> <p>1 The message object is a single message object or the last message object in a FIFO Buffer Block.</p> <p>Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
6-4	RESERVED	R	0h	Reserved

**Table 26-34. CAN\_IF1MCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	DLC	R/W	0h	Data length code 0-8 Data frame has 0-8 data bytes. 9-15 Data frame has 8 data bytes. Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message. Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**26.16.2.26 CAN\_IF1DATA Register (Offset = 110h) [reset = 0h]**

CAN\_IF1DATA is shown in [Figure 26-46](#) and described in [Table 26-35](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message. The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order. In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first. All bits in this register are write-protected by the Busy bit.

**Figure 26-46. CAN\_IF1DATA 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
Data_3								Data_2								Data_1								Data_0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 26-35. CAN\_IF1DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_3	R/W	0h	Data Byte 3 Reset type: SYSRSn
23-16	Data_2	R/W	0h	Data Byte 2 Reset type: SYSRSn
15-8	Data_1	R/W	0h	Data Byte 1 Reset type: SYSRSn
7-0	Data_0	R/W	0h	Data Byte 0 Reset type: SYSRSn

### 26.16.2.27 CAN\_IF1DATB Register (Offset = 114h) [reset = 0h]

CAN\_IF1DATB is shown in [Figure 26-47](#) and described in [Table 26-36](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message. The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order. In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first. All bits in this register are write-protected by the Busy bit.

**Figure 26-47. CAN\_IF1DATB 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
Data_7								Data_6								Data_5								Data_4							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 26-36. CAN\_IF1DATB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_7	R/W	0h	Data Byte 7 Reset type: SYSRSn
23-16	Data_6	R/W	0h	Data Byte 6 Reset type: SYSRSn
15-8	Data_5	R/W	0h	Data Byte 5 Reset type: SYSRSn
7-0	Data_4	R/W	0h	Data Byte 4 Reset type: SYSRSn



**26.16.2.28 CAN\_IF2CMD Register (Offset = 120h) [reset = 1h]**

CAN\_IF2CMD is shown in [Figure 26-48](#) and described in [Table 26-37](#).

Return to the [Summary Table](#).

The IF1/IF2 Command Registers configure and initiate the transfer between the IF1/IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred. A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 Command Register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress. After 4 to 14 clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage. If the CPU writes to both IF1/IF2 Command Registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed.

Note: While Busy bit is one, IF1/IF2 Register sets are write protected.

Note: For debug support, the auto clear functionality of the IF1/IF2 Command Registers (clear of DMAActive flag by R/W, for devices with DMA support) is disabled during Debug/Suspend mode.

Note: If an invalid Message Number is written to bits [7:0] of the IF1/IF2 Command Register, the Message Handler may access an implemented (valid) message object instead.

**Figure 26-48. CAN\_IF2CMD Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
DIR	Mask	Arb	Control	ClrIntPnd	TxRqst	DATA_A	DATA_B
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
Busy	DMAActive	RESERVED					
R-0h	R/W-0h	R-0h					
7	6	5	4	3	2	1	0
MSG_NUM							
R/W-1h							

**Table 26-37. CAN\_IF2CMD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23	DIR	R/W	0h	Write/Read  0 Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set. That is, transfer data from the mailbox into the selected IF1/IF2 Message Buffer Registers.  1 Direction = Write: Transfer direction is from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]) . That is, transfer data from the selected IF1/IF2 Message Buffer Registers to the mailbox.  The other bits of IF1/IF2 Command Mask Register have different functions depending on the transfer direction.  Note: This bit is write protected by Busy bit.  Reset type: SYSRSn

**Table 26-37. CAN\_IF2CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	Mask	R/W	0h	<p>Access Mask Bits</p> <p>0 Mask bits will not be changed</p> <p>1 (Direction = Read): The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
21	Arb	R/W	0h	<p>Access Arbitration Bits</p> <p>0 Arbitration bits will not be changed</p> <p>1 (Direction = Read): The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
20	Control	R/W	0h	<p>Access control bits.</p> <p>If the TxRqst/NewDat bit in this register(Bit [18]) is set, the TxRqst/NewDat bit in the IF1 message control register will be ignored.</p> <p>0 Control bits will not be changed.</p> <p>1 (Direction = Read): The message control bits will be transferred from the message object addressed by message number (Bits [7:0]) to the IF1 register set.</p> <p>1 (Direction = Write): The message control bits will be transferred from the IF1 register set to the message object addressed by message number (Bits [7:0]).</p> <p>Note: This bit is write protected by the Busy bit. Reset type: SYSRSn</p>
19	ClrIntPnd	R/W	0h	<p>Clear Interrupt Pending Bit</p> <p>0 IntPnd bit will not be changed</p> <p>1 (Direction = Read): Clears IntPnd bit in the message object.</p> <p>1 (Direction = Write): This bit is ignored.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>

**Table 26-37. CAN\_IF2CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	TxRqst	R/W	0h	<p>Access Transmission Request (TxRqst) / New Data (NewDat) Bit</p> <p>0 (Direction = Read): NewDat bit will not be changed.</p> <p>0 (Direction = Write): TxRqst/NewDat bit will be handled according to the Control bit.</p> <p>1 (Direction = Read): Clears NewDat bit in the message object.</p> <p>1 (Direction = Write): Sets TxRqst/NewDat in message object.</p> <p>Note: If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to one independent of the values in IF1/IF2 Message Control Register.</p> <p>Note: A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1/IF2 Message Control Register always reflect the status before resetting them.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
17	DATA_A	R/W	0h	<p>Access Data Bytes 0-3</p> <p>0 Data Bytes 0-3 will not be changed.</p> <p>1 (Direction = Read): The Data Bytes 0-3 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Data Bytes 0-3 will be transferred from the IF1/IF2 Register set to the message object addressed by the Message Number (Bits [7:0]).</p> <p>Note: The duration of the message transfer is independent of the number of bytes to be transferred.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
16	DATA_B	R/W	0h	<p>Access Data Bytes 4-7</p> <p>0 Data Bytes 4-7 will not be changed.</p> <p>1 (Direction = Read): The Data Bytes 4-7 will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Data Bytes 4-7 will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).</p> <p>Note: The duration of the message transfer is independent of the number of bytes to be transferred.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
15	Busy	R	0h	<p>Busy Flag</p> <p>0 No transfer between IF1/IF2 Register Set and Message RAM is in progress.</p> <p>1 Transfer between IF1/IF2 Register Set and Message RAM is in progress.</p> <p>This bit is set to one after the message number has been written to bits [7:0]. IF1/IF2 Register Set will be write protected. The bit is cleared after read/write action has been finished.</p> <p>Reset type: SYSRSn</p>

**Table 26-37. CAN\_IF2CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
14	DMAActive	R/W	0h	DMA trigger status due to IF1 update.  0 No IF1 DMA request is active.  1 DMA is requested after a completed transfer between IF1 and the message RAM. The DMA request remains active until the first read or write to one of the IF1 registers an exception is a write to Message Number (Bits [7:0]) when DMAActive is one.  Note: Due to the auto reset feature of the DMAActive bit, this bit has to be set for each subsequent DMA cycle separately.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
14	RESERVED	R/W	0h	Reserved
13-8	RESERVED	R	0h	Reserved
7-0	MSG_NUM	R/W	1h	Number of message object in Message RAM which is used for data transfer  0x00 Invalid message number 0x01-0x20 Valid message numbers 0x21-0xFF Invalid message numbers  Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**26.16.2.29 CAN\_IF2MSK Register (Offset = 124h) [reset = FFFFFFFFh]**

CAN\_IF2MSK is shown in [Figure 26-49](#) and described in [Table 26-38](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Mask Registers mirror the mask bits of a message object.

Note: While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write-protected.

**Figure 26-49. CAN\_IF2MSK Register**

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

**Table 26-38. CAN\_IF2MSK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MXtd	R/W	1h	Mask Extended Identifier 0 The extended identifier bit (Xtd) has no effect on the acceptance filtering. 1 The extended identifier bit (Xtd) is used for acceptance filtering. When 11-bit ("standard") identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered. Note: This bit is write protected by Busy bit. Reset type: SYRSn
30	MDir	R/W	1h	Mask Message Direction 0 The message direction bit (Dir) has no effect on the acceptance filtering. 1 The message direction bit (Dir) is used for acceptance filtering. Note: This bit is write protected by Busy bit. Reset type: SYRSn
29	RESERVED	R	1h	Reserved
28-0	Msk	R/W	1FFFFFFFh	Identifier Mask 0 The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care). 1 The corresponding bit in the identifier of the message object is used for acceptance filtering. Note: This bit is write protected by Busy bit. Reset type: SYRSn

**26.16.2.30 CAN\_IF2ARB Register (Offset = 128h) [reset = 0h]**

CAN\_IF2ARB is shown in [Figure 26-50](#) and described in [Table 26-39](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Arbitration Registers mirror the arbitration bits of a message object. The Arbitration bits ID[28:0], Xtd, and Dir are used to define the identifier and type of outgoing messages and (together with the Mask bits Msk[28:0], MXtd, and MDir) for acceptance filtering of incoming messages.

A received message is stored into the valid message object with matching identifier and Direction = receive (Data Frame) or Direction = transmit (Remote Frame).

Extended frames can be stored only in message objects with Xtd = one, standard frames in message objects with Xtd = zero.

If a received message (Data Frame or Remote Frame) matches more than one valid message objects, it is stored into the one with the lowest message number.

Note: While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write-protected.

**Figure 26-50. CAN\_IF2ARB Register**

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

**Table 26-39. CAN\_IF2ARB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MsgVal	R/W	0h	Message Valid  0 The mailbox is disabled. (The message object is ignored by the message handler). 1 The mailbox is enabled. (The message object is to be used by the message handler).  The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets the Init bit in the CAN Control Register. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
30	Xtd	R/W	0h	Extended Identifier  0 The 11-bit ("standard") Identifier is used for this message object. 1 The 29-bit ("extended") Identifier is used for this message object.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**Table 26-39. CAN\_IF2ARB Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
29	Dir	R/W	0h	<p>Message Direction</p> <p>0 Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, that frame is stored in this message object.</p> <p>1 Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
28-0	ID	R/W	0h	<p>Message Identifier</p> <p>ID[28:0] 29-bit Identifier ("Extended Frame")</p> <p>ID[28:18] 11-bit Identifier ("Standard Frame")</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>

**26.16.2.31 CAN\_IF2MCTL Register (Offset = 12Ch) [reset = 0h]**

CAN\_IF2MCTL is shown in [Figure 26-51](#) and described in [Table 26-40](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Message Control Registers mirror the message control bits of a message object. This register has control/status bits pertaining to interrupts, acceptance mask, remote frames and FIFO option.

**Figure 26-51. CAN\_IF2MCTL 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
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
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
EoB	RESERVED			DLC			
R/W-0h	R-0h			R/W-0h			

**Table 26-40. CAN\_IF2MCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	NewDat	R/W	0h	New Data  0 No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU.  1 The message handler or the CPU has written new data into the data portion of this message object.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
14	MsgLst	R/W	0h	Message Lost (only valid for message objects with direction = receive)  0 No message lost since the last time when this bit was reset by the CPU.  1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn
13	IntPnd	R/W	0h	Interrupt Pending  0 This message object is not the source of an interrupt.  1 This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.  Note: This bit is write protected by Busy bit. Reset type: SYSRSn



**Table 26-40. CAN\_IF2MCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	UMask	R/W	0h	<p>Use Acceptance Mask</p> <p>0 Mask ignored</p> <p>1 Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering</p> <p>If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
11	TxIE	R/W	0h	<p>Transmit Interrupt Enable</p> <p>0 IntPnd will not be triggered after the successful transmission of a frame.</p> <p>1 IntPnd will be triggered after the successful transmission of a frame.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
10	RxIE	R/W	0h	<p>Receive Interrupt Enable</p> <p>0 IntPnd will not be triggered after the successful reception of a frame.</p> <p>1 IntPnd will be triggered after the successful reception of a frame.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
9	RmtEn	R/W	0h	<p>Remote Enable</p> <p>0 At the reception of a remote frame, TxRqst is not changed.</p> <p>1 At the reception of a remote frame, TxRqst is set.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
8	TxRqst	R/W	0h	<p>Transmit Request</p> <p>0 This message object is not waiting for a transmission.</p> <p>1 The transmission of this message object is requested and is not yet done.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
7	EoB	R/W	0h	<p>End of Block</p> <p>0 The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.</p> <p>1 The message object is a single message object or the last message object in a FIFO Buffer Block.</p> <p>Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.</p> <p>Note: This bit is write protected by Busy bit. Reset type: SYSRSn</p>
6-4	RESERVED	R	0h	Reserved

**Table 26-40. CAN\_IF2MCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	DLC	R/W	0h	Data length code 0-8 Data frame has 0-8 data bytes. 9-15 Data frame has 8 data bytes. Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message. Note: This bit is write protected by Busy bit. Reset type: SYSRSn

### 26.16.2.32 CAN\_IF2DATA Register (Offset = 130h) [reset = 0h]

CAN\_IF2DATA is shown in [Figure 26-52](#) and described in [Table 26-41](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message. The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order. In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first. All bits in this register are write-protected by the Busy bit.

**Figure 26-52. CAN\_IF2DATA 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
Data_3								Data_2								Data_1								Data_0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 26-41. CAN\_IF2DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_3	R/W	0h	Data Byte 3 Reset type: SYSRSn
23-16	Data_2	R/W	0h	Data Byte 2 Reset type: SYSRSn
15-8	Data_1	R/W	0h	Data Byte 1 Reset type: SYSRSn
7-0	Data_0	R/W	0h	Data Byte 0 Reset type: SYSRSn

### 26.16.2.33 CAN\_IF2DATB Register (Offset = 134h) [reset = 0h]

CAN\_IF2DATB is shown in [Figure 26-53](#) and described in [Table 26-42](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message. The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order. In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first. All bits in this register are write-protected by the Busy bit.

**Figure 26-53. CAN\_IF2DATB 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
Data_7								Data_6								Data_5								Data_4							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 26-42. CAN\_IF2DATB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_7	R/W	0h	Data Byte 7 Reset type: SYSRSn
23-16	Data_6	R/W	0h	Data Byte 6 Reset type: SYSRSn
15-8	Data_5	R/W	0h	Data Byte 5 Reset type: SYSRSn
7-0	Data_4	R/W	0h	Data Byte 4 Reset type: SYSRSn

### 26.16.2.34 CAN\_IF3OBS Register (Offset = 140h) [reset = 0h]

CAN\_IF3OBS is shown in [Figure 26-54](#) and described in [Table 26-43](#).

Return to the [Summary Table](#).

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU.

The observation flags (Bits [4:0]) in the IF3 Observation register are used to determine, which data sections of the IF3 Interface Register set have to be read in order to complete a DMA read cycle. After all marked data sections are read, the DCAN is enabled to update the IF3 Interface Register set with new data.

Any access order of single bytes or half-words is supported. When using byte or half-word accesses, a data section is marked as completed, if all bytes are read.

Note: If IF3 Update Enable is used and no Observation flag is set, the corresponding message objects will be copied to IF3 without activating the DMA request line and without waiting for DMA read accesses.

A write access to this register aborts a pending DMA cycle by resetting the DMA line and enables updating of IF3 Interface Register set with new data. To avoid data inconsistency, the DMA controller should be disabled before reconfiguring IF3 observation register. The status of the current read-cycle can be observed via status flags (Bits [12:8]).

With this, the observation status bits and the IF3Upd bit could be used by the application to realize the notification about new IF3 content in polling or interrupt mode

**Figure 26-54. CAN\_IF3OBS 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
IF3Upd	RESERVED		IF3SDB	IF3SDA	IF3SC	IF3SA	IF3SM
R-0h	R-0h		R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED			Data_B	Data_A	Ctrl	Arb	Mask
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 26-43. CAN\_IF3OBS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	IF3Upd	R	0h	IF3 Update Data  0 No new data has been loaded since last IF3 read. 1 New data has been loaded since last IF3 read. Reset type: SYSRSn
14-13	RESERVED	R	0h	Reserved
12	IF3SDB	R	0h	IF3 Status of Data B read access  0 All Data B bytes are already read out, or are not marked to be read. 1 Data B section has still data to be read out. Reset type: SYSRSn

**Table 26-43. CAN\_IF3OBS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	IF3SDA	R	0h	IF3 Status of Data A read access  0 All Data A bytes are already read out, or are not marked to be read.  1 Data A section has still data to be read out. Reset type: SYSRSn
10	IF3SC	R	0h	IF3 Status of Control bits read access  0 All Control section bytes are already read out, or are not marked to be read.  1 Control section has still data to be read out. Reset type: SYSRSn
9	IF3SA	R	0h	IF3 Status of Arbitration data read access  0 All Arbitration data bytes are already read out, or are not marked to be read.  1 Arbitration section has still data to be read out. Reset type: SYSRSn
8	IF3SM	R	0h	IF3 Status of Mask data read access  0 All Mask data bytes are already read out, or are not marked to be read.  1 Mask section has still data to be read out. Reset type: SYSRSn
7-5	RESERVED	R	0h	Reserved
4	Data_B	R/W	0h	Data B read observation  0 Data B section not to be read.  1 Data B section has to be read to enable next IF3 update. Reset type: SYSRSn
3	Data_A	R/W	0h	Data A read observation  0 Data A section not to be read.  1 Data A section has to be read to enable next IF3 update. Reset type: SYSRSn
2	Ctrl	R/W	0h	Ctrl read observation  0 Ctrl section not to be read.  1 Ctrl section has to be read to enable next IF3 update. Reset type: SYSRSn
1	Arb	R/W	0h	Arbitration data read observation  0 Arbitration data not to be read.  1 Arbitration data has to be read to enable next IF3 update. Reset type: SYSRSn
0	Mask	R/W	0h	Mask data read observation  0 Mask data not to be read.  1 Mask data has to be read to enable next IF3 update. Reset type: SYSRSn

**26.16.2.35 CAN\_IF3MSK Register (Offset = 144h) [reset = FFFFFFFFh]**

CAN\_IF3MSK is shown in [Figure 26-55](#) and described in [Table 26-44](#).

Return to the [Summary Table](#).

This register provides a window to the acceptance mask for the chosen mailbox.

**Figure 26-55. CAN\_IF3MSK Register**

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

**Table 26-44. CAN\_IF3MSK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MXtd	R	1h	Mask Extended Identifier  0 The extended identifier bit (Xtd) has no effect on the acceptance filtering.  1 The extended identifier bit (Xtd) is used for acceptance filtering.  Note: When 11-bit ("standard") identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.  Reset type: SYSRSn
30	MDir	R	1h	Mask Message Direction  0 The message direction bit (Dir) has no effect on the acceptance filtering.  1 The message direction bit (Dir) is used for acceptance filtering.  Reset type: SYSRSn
29	RESERVED	R	1h	Reserved
28-0	Msk	R	1FFFFFFFh	Identifier Mask Identifier Mask  0 The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).  1 The corresponding bit in the identifier of the message object is used for acceptance filtering. Identifier Mask  Reset type: SYSRSn

**26.16.2.36 CAN\_IF3ARB Register (Offset = 148h) [reset = 0h]**

CAN\_IF3ARB is shown in [Figure 26-56](#) and described in [Table 26-45](#).

Return to the [Summary Table](#).

The bits of the IF3 Arbitration Register mirrors the arbitration bits of a message object.

**Figure 26-56. CAN\_IF3ARB Register**

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

**Table 26-45. CAN\_IF3ARB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MsgVal	R	0h	Message Valid  0 The message object is ignored by the message handler. 1 The message object is to be used by the message handler.  The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required. Reset type: SYSRSn
30	Xtd	R	0h	Extended Identifier  0 The 11-bit ("standard") Identifier is used for this message object. 1 The 29-bit ("extended") Identifier is used for this message object. Reset type: SYSRSn
29	Dir	R	0h	Message Direction  0 Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, that message is stored in this message object.  1 Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one). Reset type: SYSRSn
28-0	ID	R	0h	Message Identifier  ID[28:0] 29-bit Identifier ("Extended Frame") ID[28:18] 11-bit Identifier ("Standard Frame") Reset type: SYSRSn



### 26.16.2.37 CAN\_IF3MCTL Register (Offset = 14Ch) [reset = 0h]

CAN\_IF3MCTL is shown in [Figure 26-57](#) and described in [Table 26-46](#).

Return to the [Summary Table](#).

The bits of the IF3 Message Control Register mirrors the message control bits of a message object.

**Figure 26-57. CAN\_IF3MCTL 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
NewDat	MsgLst	IntPnd	UMask	TxE	RxE	RmtEn	TxRqst
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
EoB	RESERVED				DLC		
R-0h	R-0h				R-0h		

**Table 26-46. CAN\_IF3MCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	NewDat	R	0h	New Data  0 No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU.  1 The message handler or the CPU has written new data into the data portion of this message object. Reset type: SYSRSn
14	MsgLst	R	0h	Message Lost (only valid for message objects with direction = receive)  0 No message lost since the last time when this bit was reset by the CPU.  1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten. Reset type: SYSRSn
13	IntPnd	R	0h	Interrupt Pending  0 This message object is not the source of an interrupt.  1 This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority. Reset type: SYSRSn
12	UMask	R	0h	Use Acceptance Mask  0 Mask ignored  1 Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering  If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one. Reset type: SYSRSn

**Table 26-46. CAN\_IF3MCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	TxE	R	0h	<p>Transmit Interrupt Enable</p> <p>0 IntPnd will not be triggered after the successful transmission of a frame.</p> <p>1 IntPnd will be triggered after the successful transmission of a frame.</p> <p>Reset type: SYSRSn</p>
10	RxE	R	0h	<p>Receive Interrupt Enable</p> <p>0 IntPnd will not be triggered after the successful reception of a frame.</p> <p>1 IntPnd will be triggered after the successful reception of a frame.</p> <p>Reset type: SYSRSn</p>
9	RmtEn	R	0h	<p>Remote Enable</p> <p>0 At the reception of a remote frame, TxRqst is not changed.</p> <p>1 At the reception of a remote frame, TxRqst is set.</p> <p>Reset type: SYSRSn</p>
8	TxRqst	R	0h	<p>Transmit Request</p> <p>0 This message object is not waiting for a transmission.</p> <p>1 The transmission of this message object is requested and is not yet done.</p> <p>Reset type: SYSRSn</p>
7	EoB	R	0h	<p>End of Block</p> <p>0 The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.</p> <p>1 The message object is a single message object or the last message object in a FIFO Buffer Block.</p> <p>Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.</p> <p>Reset type: SYSRSn</p>
6-4	RESERVED	R	0h	Reserved
3-0	DLC	R	0h	<p>Data length code</p> <p>0-8 Data frame has 0-8 data bytes.</p> <p>9-15 Data frame has 8 data bytes.</p> <p>Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.</p> <p>Reset type: SYSRSn</p>

**26.16.2.38 CAN\_IF3DATA Register (Offset = 150h) [reset = 0h]**

CAN\_IF3DATA is shown in [Figure 26-58](#) and described in [Table 26-47](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message.

**Figure 26-58. CAN\_IF3DATA 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
Data_3								Data_2								Data_1								Data_0							
R-0h								R-0h								R-0h								R-0h							

**Table 26-47. CAN\_IF3DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_3	R	0h	Data Byte 3 Reset type: SYSRSn
23-16	Data_2	R	0h	Data Byte 2 Reset type: SYSRSn
15-8	Data_1	R	0h	Data Byte 1 Reset type: SYSRSn
7-0	Data_0	R	0h	Data Byte 0 Reset type: SYSRSn

**26.16.2.39 CAN\_IF3DATB Register (Offset = 154h) [reset = 0h]**

CAN\_IF3DATB is shown in [Figure 26-59](#) and described in [Table 26-48](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message.

**Figure 26-59. CAN\_IF3DATB 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
Data_7								Data_6								Data_5								Data_4							
R-0h								R-0h								R-0h								R-0h							

**Table 26-48. CAN\_IF3DATB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_7	R	0h	Data Byte 7 Reset type: SYSRSn
23-16	Data_6	R	0h	Data Byte 6 Reset type: SYSRSn
15-8	Data_5	R	0h	Data Byte 5 Reset type: SYSRSn
7-0	Data_4	R	0h	Data Byte 4 Reset type: SYSRSn

**26.16.2.40 CAN\_IF3UPD Register (Offset = 160h) [reset = 0h]**

CAN\_IF3UPD is shown in [Figure 26-60](#) and described in [Table 26-49](#).

Return to the [Summary Table](#).

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set. Note: IF3 Update enable should not be set for transmit objects.

**Figure 26-60. CAN\_IF3UPD 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
IF3UpdEn																															
R/W-0h																															

**Table 26-49. CAN\_IF3UPD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	IF3UpdEn	R/W	0h	<p>IF3 Update Enabled (for all message objects)</p> <p>0 Automatic IF3 update is disabled for this message object.</p> <p>1 Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.</p> <p>Reset type: SYSRSn</p>

### 26.16.3 Register to Driverlib Function Mapping

**Table 26-50. CAN Registers to Driverlib Functions**

File	Driverlib Function
<b>CTL</b>	
can.c	CAN_initModule
can.c	CAN_setBitTiming
can.h	CAN_startModule
can.h	CAN_enableController
can.h	CAN_disableController
can.h	CAN_enableTestMode
can.h	CAN_disableTestMode
can.h	CAN_setInterruptionDebugMode
can.h	CAN_enableDMARRequests
can.h	CAN_disableDMARRequests
can.h	CAN_disableAutoBusOn
can.h	CAN_enableAutoBusOn
can.h	CAN_enableInterrupt
can.h	CAN_disableInterrupt
can.h	CAN_enableRetry
can.h	CAN_disableRetry
can.h	CAN_isRetryEnabled
<b>ES</b>	
can.c	CAN_clearInterruptStatus
can.h	CAN_getStatus
<b>ERRC</b>	
can.h	CAN_getErrorCount
<b>BTR</b>	
can.c	CAN_setBitTiming
can.h	CAN_getBitTiming
<b>INT</b>	
can.h	CAN_getInterruptCause
<b>TEST</b>	
can.h	CAN_enableTestMode
can.h	CAN_disableTestMode
can.h	CAN_enableMemoryAccessMode
can.h	CAN_disableMemoryAccessMode
<b>RAM_INIT</b>	
can.h	CAN_initRAM
<b>GLB_INT_EN</b>	
can.h	CAN_enableGlobalInterrupt
can.h	CAN_disableGlobalInterrupt
<b>GLB_INT_FLG</b>	
can.h	CAN_getGlobalInterruptStatus
<b>GLB_INT_CLR</b>	
can.h	CAN_clearGlobalInterruptStatus
<b>ABOTR</b>	
can.h	CAN_setAutoBusOnTime
<b>TXRQ_21</b>	
can.h	CAN_getTxRequests

**Table 26-50. CAN Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>NDAT_21</b>	
can.h	CAN_getNewDataFlags
<b>IPEN_21</b>	
can.h	CAN_getInterruptMessageSource
<b>MVAL_21</b>	
can.h	CAN_getValidMessageObjects
<b>IP_MUX21</b>	
can.h	CAN_getInterruptMux
can.h	CAN_setInterruptMux
<b>IF1CMD</b>	
can.c	CAN_clearInterruptStatus
can.c	CAN_setupMessageObject
can.c	CAN_sendMessage
can.c	CAN_transferMessage
can.c	CAN_clearMessage
<b>IF1MSK</b>	
can.c	CAN_setupMessageObject
<b>IF1ARB</b>	
can.c	CAN_setupMessageObject
can.c	CAN_clearMessage
<b>IF1MCTL</b>	
can.c	CAN_setupMessageObject
can.c	CAN_sendMessage
<b>IF1DATA</b>	
can.c	CAN_sendMessage
<b>IF1DATB</b>	
-	See IF1DATA
<b>IF2CMD</b>	
can.c	CAN_readMessage
can.c	CAN_transferMessage
<b>IF2MCTL</b>	
can.c	CAN_readMessage
<b>IF2DATA</b>	
can.c	CAN_readMessage
<b>IF2DATB</b>	
-	See IF2DATA
<b>IF3DATB</b>	
-	See IF3DATA

## **Local Interconnect Network (LIN)**

---



---

This chapter describes the local interconnect network (LIN) module. Since this module can also operate like a conventional serial communications interface (SCI) port, it is referred to as the SCI/LIN module in this document. In SCI compatibility mode, it is functionally compatible to the standalone SCI module. However, since the SCI/LIN module uses a different register/bit structure, code written for this module cannot be directly ported to the standalone SCI module and vice versa.

This module can be configured to operate in either SCI (UART) or LIN mode.

Topic	Page
<b>27.1 Introduction .....</b>	<b>2473</b>
<b>27.2 Serial Communications Interface Module.....</b>	<b>2478</b>
<b>27.3 Local Interconnect Network Module .....</b>	<b>2495</b>
<b>27.4 Low-Power Mode .....</b>	<b>2516</b>
<b>27.5 Emulation Mode.....</b>	<b>2518</b>
<b>27.6 SCI/LIN Registers.....</b>	<b>2519</b>



## 27.1 Introduction

The SCI/LIN is compliant to the LIN 2.1 protocol specified in the *LIN Specification Package*. The SCI/LIN module can be programmed to work either as an SCI or as a LIN. The SCI's hardware features are augmented to achieve LIN compatibility.

The SCI module is a universal asynchronous receiver-transmitter (UART) that implements the standard non-return to zero format.

The LIN standard is based on the SCI (UART) serial data link format. The communication concept is single-master/multiple-slave with a message identification for multi-cast transmission between any network nodes.

Throughout the chapter, Compatibility Mode refers to SCI Mode functionality of the SCI/LIN Module. [Section 27.2](#) explains about the SCI functionality and [Section 27.3](#) explains about the LIN functionality. Though the registers are common for LIN and SCI, the register descriptions has notes to identify the register/bit usage in different modes.

### 27.1.1 SCI Features

The following are the features of the SCI module:

- Standard universal asynchronous receiver-transmitter (UART) communication
- Supports full- or half-duplex operation
- Standard non-return to zero (NRZ) format
- Double-buffered receive and transmit functions in compatibility mode
- Supports two individually enabled interrupt lines: level 0 and level 1
- Configurable frame format of 3 to 13 bits per character based on the following:
  - Data word length programmable from one to eight bits
  - Additional address bit in address-bit mode
  - Parity programmable for zero or one parity bit, odd or even parity
  - Stop programmable for one or two stop bits
- Asynchronous communication mode
- Two multiprocessor communication formats allow communication between more than two devices
- Sleep mode is available to free CPU resources during multiprocessor communication and then wake up to receive an incoming message
- The 24-bit programmable baud rate supports  $2^{24}$  different baud rates provide high accuracy baud rate selection
- At 100-MHz peripheral clock, 3.125 Mbits/s is the Max Baud Rate achievable
- Capability to use Direct Memory Access (DMA) for transmit and receive data
- Five error flags and Seven status flags provide detailed information regarding SCI events
- Two external pins: LINRX and LINTX
- Multi-buffered receive and transmit units

---

**NOTE:**

- The SCI/LIN module is functionally compatible with C2000 SCI modules, but not directly software compatible due to different register control structures.
  - The SCI/LIN module does not support UART hardware flow control. This feature can be implemented in software using a general purpose I/O pin.
  - The SCI/LIN module does not support isosynchronous mode as there is no SCICLK pin.
-

### 27.1.2 LIN Features

The following are the features of the LIN module:

- Compatibility with LIN 1.3, 2.0, and 2.1 protocols
- Configurable Baud Rate up to 20 kpbs
- Two external pins: LINRX and LINTX.
- Multi-buffered receive and transmit units
- Identification masks for message filtering
- Automatic master header generation
  - Programmable synchronization break field
  - Synchronization field
  - Identifier field
- Slave automatic synchronization
  - Synchronization break detection
  - Optional baud rate update
  - Synchronization validation
- $2^{31}$  programmable transmission rates with 7 fractional bits
- Wake up on LINRX dominant level from transceiver
- Automatic wake up support
  - Wakeup signal generation
  - Expiration times on wakeup signals
- Automatic bus idle detection
- Error detection
  - Bit error
  - Bus error
  - No-response error
  - Checksum error
  - Synchronization field error
  - Parity error
- Capability to use Direct Memory Access (DMA) for transmit and receive data.
- 2 Interrupt lines with priority encoding for:
  - Receive
  - Transmit
  - ID, error, and status
- Support for LIN 2.0 checksum
- Enhanced synchronizer finite state machine (FSM) support for frame processing
- Enhanced handling of extended frames
- Enhanced baud rate generator
- Update wakeup/go to sleep

### 27.1.3 Block Diagram

The SCI/LIN module contains core SCI block with added sub-blocks to support LIN protocol.

Three Major components of the SCI Module are:

- Transmitter
- Baud Clock Generator
- Receiver

**Transmitter (TX)** contains two major registers to perform the double- buffering:

- The transmitter data buffer register (SCITD) contains data loaded by the CPU to be transferred to the shift register for transmission.
- The transmitter shift register (SCITXSHF) loads data from the data buffer (SCITD) and shifts data onto the LINTX pin, one bit at a time.

#### **Baud Clock Generator**

- A programmable baud generator produces a baud clock scaled from the input clock VCLK

**Receiver (RX)** contains two major registers to perform the double- buffering:

- The receiver shift register (SCIRXSHF) shifts data in from the LINRX pin one bit at a time and transfers completed data into the receive data buffer.
- The receiver data buffer register (SCIRD) contains received data transferred from the receiver shift register

The SCI receiver and transmitter are double-buffered, and each has its own separate enable and interrupt bits. The receiver and transmitter may each be operated independently or simultaneously in full duplex mode.

To ensure data integrity, the SCI checks the data it receives for breaks, parity, overrun, and framing errors. The bit rate (baud) is programmable to over 16 million different rates through a 24-bit baud-select register. [Figure 27-1](#) shows the detailed SCI block diagram.

The SCI/LIN module is based on the standalone SCI with the addition of an error detector (parity calculator, checksum calculator, and bit monitor), a mask filter, a synchronizer, and a multi-buffered receiver and transmitter. The SCI interface, the DMA control subblocks and the baud generator are modified as part of the hardware enhancements for LIN compatibility. [Figure 27-2](#) shows the SCI/LIN block diagram.

Figure 27-1. SCI Block Diagram

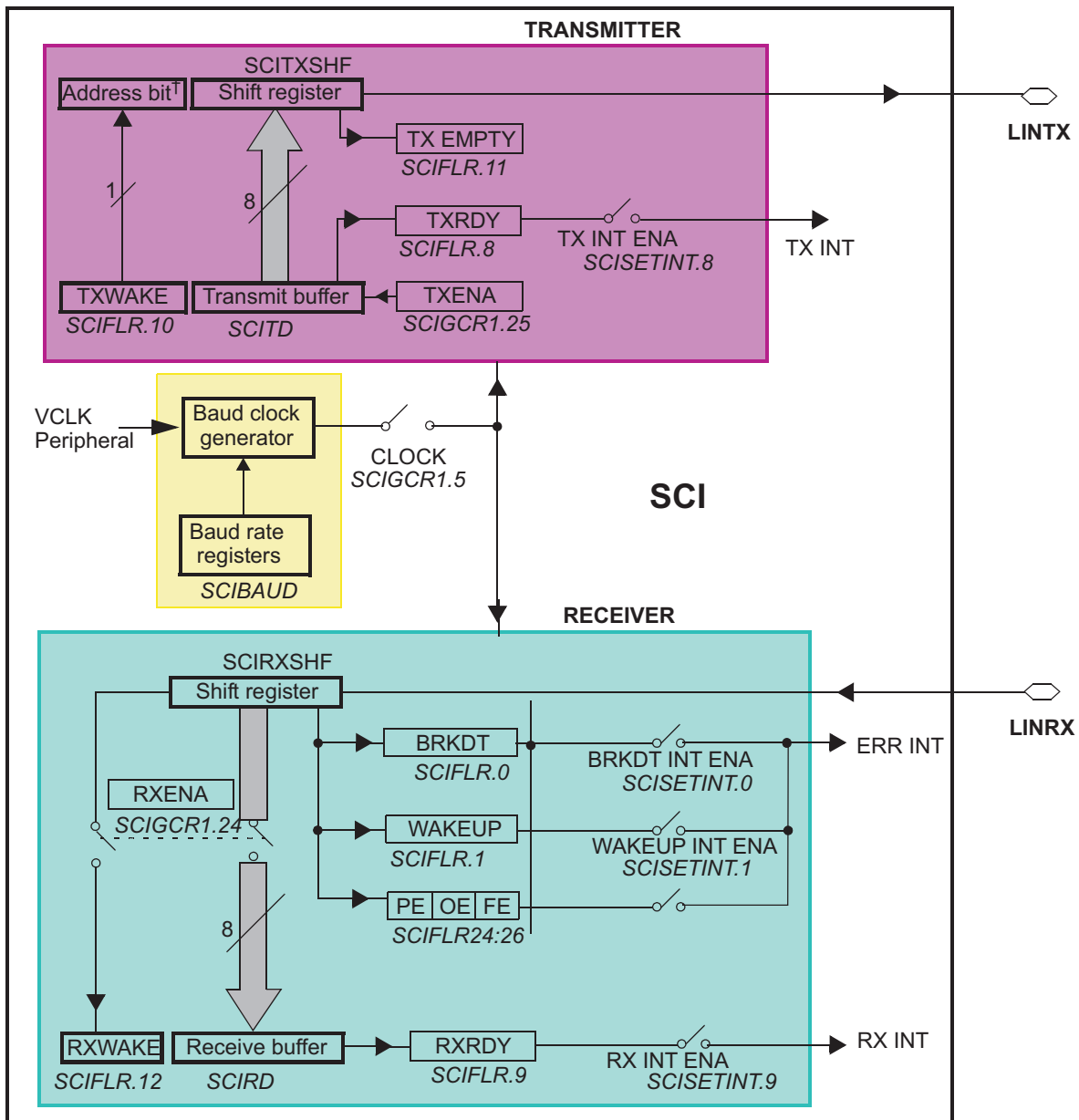
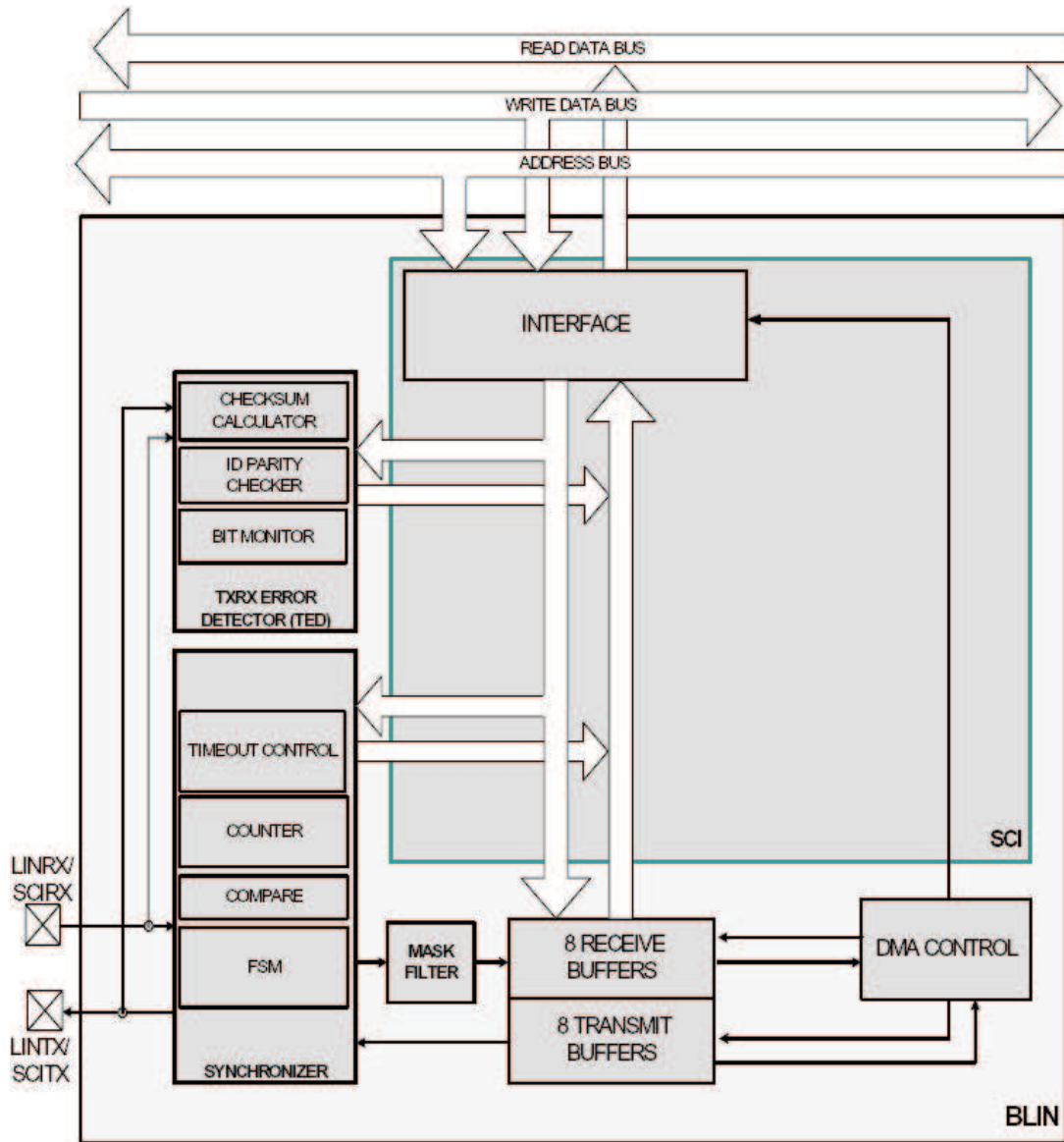


Figure 27-2. SCI/LIN Block Diagram



## 27.2 Serial Communications Interface Module

### 27.2.1 SCI Communication Formats

The SCI module can be configured to meet the requirements of many applications. Because communication formats vary depending on the specific application, many attributes of the SCI/LIN are user configurable. The list below describes these configuration options:

- SCI Frame format
- SCI Timing modes
- SCI Baud rate
- SCI Multiprocessor modes

#### 27.2.1.1 SCI Frame Formats

The SCI uses a programmable frame format. All frames consist of the following:

- One start bit
- One to eight data bits
- Zero or one address bit
- Zero or one parity bit
- One or two stop bits

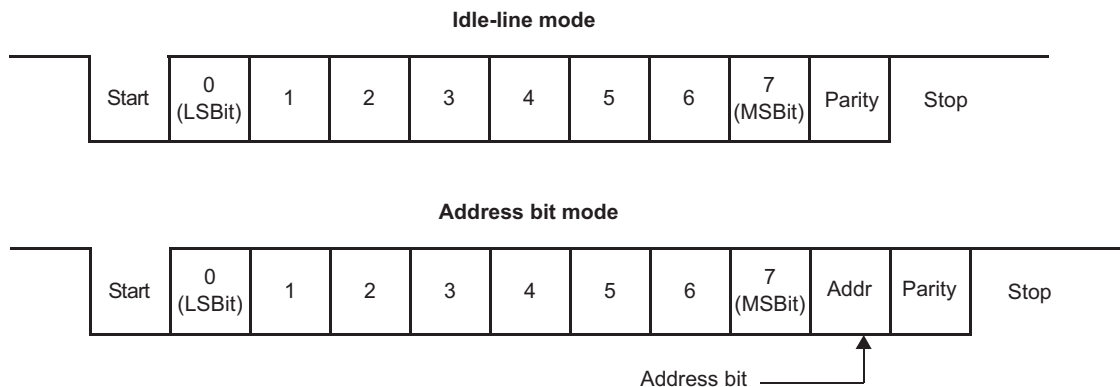
The frame format for both the transmitter and receiver is programmable through the bits in the SCIGCR1 register. Both receive and transmit data is in nonreturn to zero (NRZ) format, which means that the transmit and receive lines are at logic high when idle. Each frame transmission begins with a start bit, in which the transmitter pulls the SCI line low (logic low). Following the start bit, the frame data is sent and received least significant bit first (LSB).

An address bit is present in each frame if the SCI is configured to be in address-bit mode but is not present in any frame if the SCI is configured for idle-line mode. The format of frames with and without the address bit is illustrated in [Figure 27-3](#).

A parity bit is present in every frame when the PARITY ENA bit is set. The value of the parity bit depends on the number of one bits in the frame and whether odd or even parity has been selected via the PARITY ENA bit. Both examples in [Figure 27-3](#) have parity enabled.

All frames include one stop bit, which is always a high level. This high level at the end of each frame is used to indicate the end of a frame to ensure synchronization between communicating devices. Two stop bits are transmitted if the STOP bit in SCIGCR1 register is set. The examples shown in [Figure 27-3](#) use one stop bit per frame.

**Figure 27-3. Typical SCI Data Frame Formats**



### 27.2.1.2 SCI Timing Mode

The SCI can be configured to use asynchronous timing mode using TIMING MODE bit in SCIGCR1 register.

#### 27.2.1.2.1 Asynchronous Timing Mode

The asynchronous timing mode uses only the receive and transmit data lines to interface with devices using the standard universal asynchronous receiver-transmitter (UART) protocol.

In the asynchronous timing mode, each bit in a frame has a duration of 16 SCI baud clock periods. Each bit therefore consists of 16 samples (one for each clock period). When the SCI is using asynchronous mode, the baud rates of all communicating devices must match as closely as possible. Receive errors result from devices communicating at different baud rates.

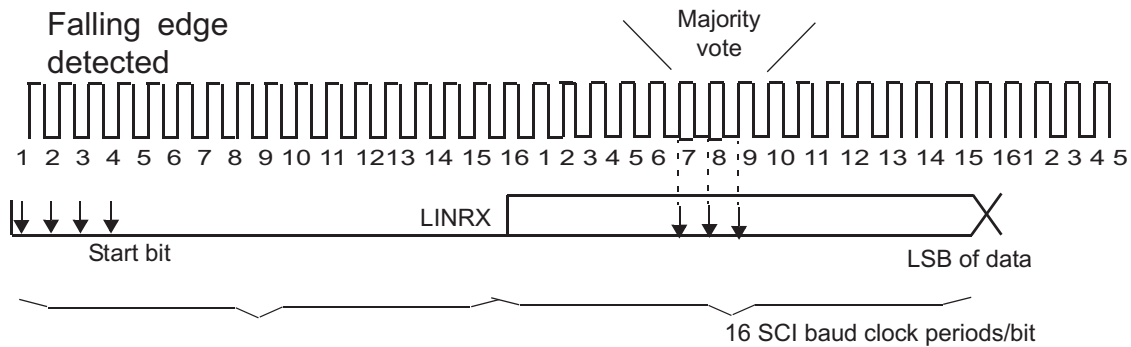
With the receiver in the asynchronous timing mode, the SCI detects a valid start bit if the first four samples after a falling edge on the LINRX pin are of logic level 0. As soon as a falling edge is detected on LINRX, the SCI assumes that a frame is being received and synchronizes itself to the bus.

To prevent interpreting noise as Start bit SCI expects LINRX line to be low for at least four contiguous SCI baud clock periods to detect a valid start bit. The bus is considered idle if this condition is not met. When a valid start bit is detected, the SCI determines the value of each bit by sampling the LINRX line value during the seventh, eighth, and ninth SCI baud clock periods. A majority vote of these three samples is used to determine the value stored in the SCI receiver shift register. By sampling in the middle of the bit, the SCI reduces errors caused by propagation delays and rise and fall times and data line noises.

Figure 27-4 illustrates how the receiver samples a start bit and a data bit in asynchronous timing mode.

The transmitter transmits each bit for a duration of 16 SCI baud clock periods. During the first clock period for a bit, the transmitter shifts the value of that bit onto the LINTX pin. The transmitter then holds the current bit value on LINTX for 16 SCI baud clock periods.

Figure 27-4. Asynchronous Communication Bit Timing



### 27.2.1.3 SCI Baud Rate

The SCI/LIN has an internally generated serial clock determined by the peripheral VCLK and the prescalers P and M in this register. The SCI uses the 24-bit integer prescaler P value in the BRS register to select the required baud rates. The additional 4-bit fractional divider M refines the baud rate selection.

In asynchronous timing mode, the SCI generates a baud clock according to the following formula:

$$SCICLK \text{ Frequency} = \frac{VCLK \text{ Frequency}}{P + 1 + \frac{M}{16}}$$

$$\text{Asynchronous baud value} = \frac{SCICLK \text{ Frequency}}{16}$$

For P = 0,

$$\text{Asynchronous baud value} = \frac{VCLK \text{ Frequency}}{32} \quad (13)$$

**Table 27-1. P and M values for standard bit-rates<sup>(1)</sup>**

Desired bit rate	Actual bit-rate	Percentage error	P value	M value
115200	115207.4	0.0064	26	2
57600	57603.69	0.0064	53	4
38400	38402.46	0.0064	80	6
19200	19201.23	0.0064	161	12
9600	9600.61	0.0064	324	8
4800	4799.85	0.0032	650	1
2400	2400.04	0.0016	1301	1
1200	1199.99	0.0008	2603	1

<sup>(1)</sup> For a 100 MHz device, LIN module input clock = 50 MHz

### 27.2.1.3.1 Superfractional Divider, SCI Asynchronous Mode

The superfractional divider is available in SCI asynchronous mode (idle-line and address-bit mode). Building on the 4-bit fractional divider M (BRS[27:24]), the superfractional divider uses an additional 3-bit modulating value (see [Table 27-3](#)). The bits with a 1 in the table will have an additional VCLK period added to their  $T_{bit}$ . If the character length is more than 10, then the modulation table will be a rolled-over version of the original table ([Table 27-2](#)), as shown in [Table 27-3](#).

The baud rate will vary over a data field to average according to the BRS[30:28] value by a “d” fraction of the peripheral internal clock:  $0 < d < 1$ . See [Figure 27-5](#) for a simple Average “d” calculation based on “U” value (BRS[30:28]).

The instantaneous bit time is expressed in terms of  $T_{VCLK}$  as follows:

For all P other than 0, and all M and d (0 or 1),

$$T^{i}bit = \left[ 16 \left( P + 1 + \frac{M}{16} \right) + d \right] T_{VCLK} \quad (14)$$

For P = 0  $T_{bit} = 32T_{VCLK}$

The averaged bit time is expressed in terms of  $T_{VCLK}$  as follows:

For all P other than 0, and all M and d ( $0 < d < 1$ ),

$$T^{a}bit = \left[ 16 \left( P + 1 + \frac{M}{16} \right) + d \right] T_{VCLK} \quad (15)$$

For P = 0  $T_{bit} = 32T_{VCLK}$



**Table 27-2. Superfractional Bit Modulation for SCI Mode (Normal Configuration) <sup>(1)</sup>**

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0h	0	0	0	0	0	0	0	0	0	0
1h	1	0	0	0	0	0	0	0	1	0
2h	1	0	0	0	1	0	0	0	1	0
3h	1	0	1	0	1	0	0	0	1	0
4h	1	0	1	0	1	0	1	0	1	0
5h	1	1	1	0	1	0	1	0	1	1
6h	1	1	1	0	1	1	1	0	1	1
7h	1	1	1	1	1	1	1	0	1	1

<sup>(1)</sup> Normal configuration = Start + 8 Data Bits + Stop Bit

**Table 27-3. Superfractional Bit Modulation for SCI Mode (Maximum Configuration) <sup>(1)</sup>**

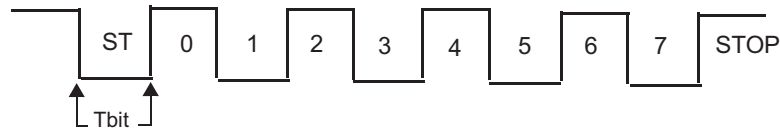
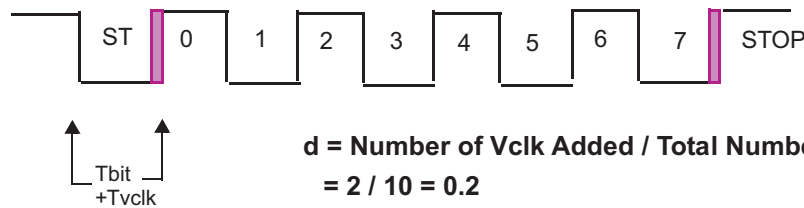
BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Addr	Parity	Stop0	Stop1
0h	0	0	0	0	0	0	0	0	0	0	0	0	0
1h	1	0	0	0	0	0	0	0	1	0	0	0	0
2h	1	0	0	0	1	0	0	0	1	0	0	0	1
3h	1	0	1	0	1	0	0	0	1	0	1	0	1
4h	1	0	1	0	1	0	1	0	1	0	1	0	1
5h	1	1	1	0	1	0	1	0	1	1	1	0	1
6h	1	1	1	0	1	1	1	0	1	1	1	0	1
7h	1	1	1	1	1	1	1	0	1	1	1	1	1

<sup>(1)</sup> Maximum configuration = Start + 8 Data Bits + Addr Bit + Parity Bit + Stop Bit 0 + Stop Bit 1

**Table 27-4. SCI Mode (Minimum Configuration) <sup>(1)</sup>**

BRS[30:28]	Start Bit	D[0]	Stop
0h	0	0	0
1h	1	0	0
2h	1	0	0
3h	1	0	1
4h	1	0	1
5h	1	1	1
6h	1	1	1
7h	1	1	1

<sup>(1)</sup> Minimum configuration = Start + 1 Data Bits + Stop Bit

**Figure 27-5. Superfractional Divider Example**
**Normal Data Frame with BRS[31:28] = 0**

**Normal Data Frame with BRS[31:28] = 1**

**Table 27-5. Comparative Baud Values for Different P Values, Asynchronous Mode<sup>(1)(2)</sup>**

24-Bit Register Value		Baud Selected		Percent Error
Decimal	Hex	Ideal	Actual	
26	00001A	115200	115740	0.47
53	000035	57600	57870	0.47
80	000050	38400	38580	0.47
162	0000A2	19200	19172	-0.15
299	00012B	10400	10417	0.16
325	000145	9600	9586	-0.15
399	00018F	7812.5	7812.5	0.00
650	00028A	4800	4800	0.00
15624	003BA0	200	200	0.00
624999	098967	5	5	0.00

<sup>(1)</sup> VCLK = 50 MHz

<sup>(2)</sup> Values are in decimal except for column 2.

### 27.2.1.4 SCI Multiprocessor Communication Modes

In some applications, the SCI may be connected to more than one serial communication device. In such a multiprocessor configuration, several frames of data may be sent to all connected devices or to an individual device. In the case of data sent to an individual device, the receiving devices must determine when they are being addressed. When a message is not intended for them, the devices can ignore the following data. When only two devices make up the SCI network, addressing is not needed, so multiprocessor communication schemes are not required.

SCI supports two multiprocessor communication modes which can be selected using COMM MODE bit:

- Idle-Line Mode
- Address Bit Mode

When the SCI is not used in a multiprocessor environment, software can consider all frames as data frames. In this case, the only distinction between the idle-line and address-bit modes is the presence of an extra bit (the address bit) in each frame sent with the address-bit protocol.

The SCI allows full-duplex communication where data can be sent and received via the transmit and receive pins simultaneously. However, the protocol used by the SCI assumes that only one device transmits data on the same bus line at any one time. No arbitration is done by the SCI.

#### 27.2.1.4.1 Idle-Line Multiprocessor Modes

In idle-line multiprocessor mode, a frame that is preceded by an idle period (10 or more idle bits) is an address frame. A frame that is preceded by fewer than 10 idle bits is a data frame. [Figure 27-6](#) illustrates the format of several blocks and frames with idle-line mode.

There are two ways to transmit an address frame using idle-line mode:

**Method 1:** In software, deliberately leave an idle period between the transmission of the last data frame of the previous block and the address frame of the new block.

**Method 2:** Configure the SCI to automatically send an idle period between the last data frame of the previous block and the address frame of the new block.

Although Method 1 is only accomplished by a delay loop in software, Method 2 can be implemented by using the transmit buffer and the TXWAKE bit in the following manner:

Step 1: Write a 1 to the TXWAKE bit.

Step 2: Write a dummy data value to the SCITD register. This triggers the SCI to begin the idle period as soon as the transmitter shift register is empty.

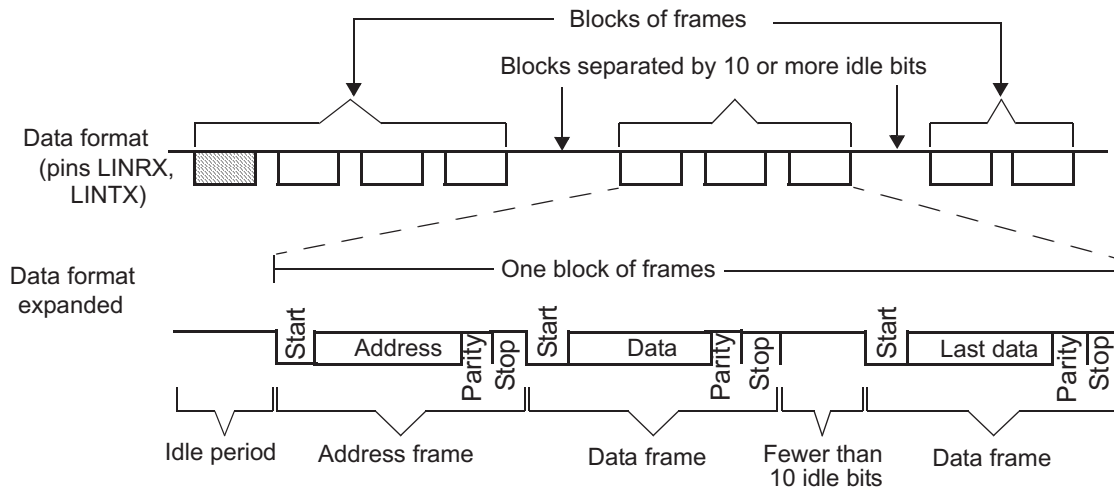
Step 3: Wait for the SCI to clear the TXWAKE flag.

Step 4: Write the address value to SCITD.

As indicated by Step 3, software should wait for the SCI to clear the TXWAKE bit. However, the SCI clears the TXWAKE bit at the same time it sets TXRDY (that is, transfers data from SCITD into SCITXSHF). Therefore, if the TX INT ENA bit is set, the transfer of data from SCITD to SCITXSHF causes an interrupt to be generated at the same time that the SCI clears the TXWAKE bit. If this interrupt method is used, software is not required to poll the TXWAKE bit waiting for the SCI to clear it.

When idle-line multiprocessor communications are used, software must ensure that the idle time exceeds 10 bit periods before addresses (using one of the methods mentioned above), and software must also ensure that data frames are written to the transmitter quickly enough to be sent without a delay of 10 bit periods between frames. Failure to comply with these conditions will result in data interpretation errors by other devices receiving the transmission.

**Figure 27-6. Idle-Line Multiprocessor Communication Format**



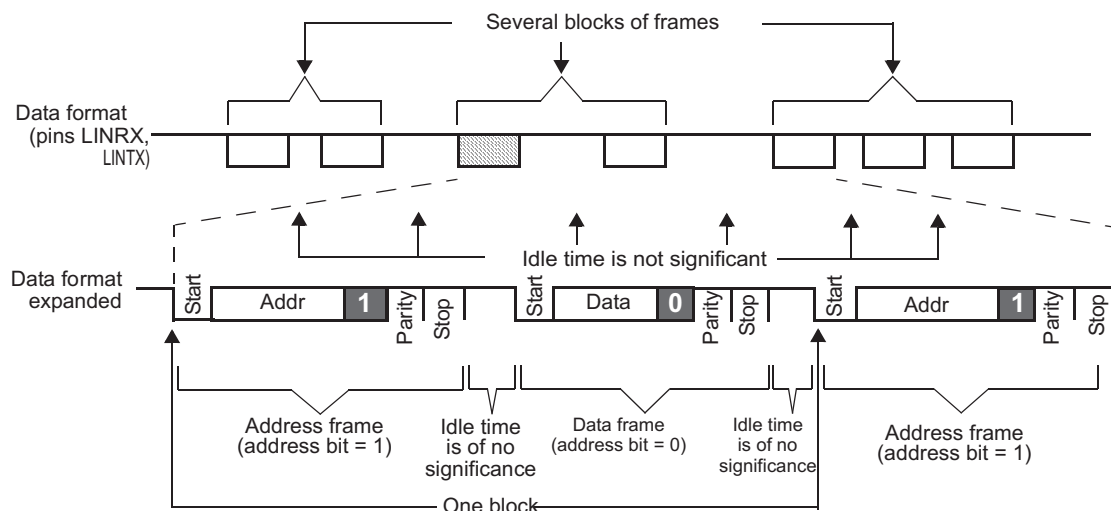
**27.2.1.4.2 Address-Bit Multiprocessor Mode**

In the address-bit protocol, each frame has an extra bit immediately following the data field called an address bit. A frame with the address bit set to 1 is an address frame; a frame with the address bit set to 0 is a data frame. The idle period timing is irrelevant in this mode. [Figure 27-7](#) illustrates the format of several blocks and frames with the address-bit mode.

When address-bit mode is used, the value of the TXWAKE bit is the value sent as the address bit. To send an address frame, software must set the TXWAKE bit. This bit is cleared as the contents of the SCITD are shifted from the TXWAKE register so that all frames sent are data except when the TXWAKE bit is written as a 1.

No dummy write to SCITD is required before an address frame is sent in address-bit mode. The first byte written to SCITD after the TXWAKE bit is written to 1 is transmitted with the address bit set when address-bit mode is used.

**Figure 27-7. Address-Bit Multiprocessor Communication Format**



### 27.2.1.5 SCI Multi Buffered Mode

To reduce CPU load when Receiving or Transmitting data in interrupt mode or DMA mode, the SCI/LIN module has eight separate receive and transmit buffers. Multi buffered mode is enabled by setting the MBUF MODE bit.

The multi-buffer 3-bit counter counts the data bytes transferred from the SCIRXSHF register to the RDy receive buffers and TDy transmit buffers register to SCITXSHF register. The 3-bit compare register contains the number of data bytes expected to be received or transmitted. the LENGTH value in SCIFORMAT register indicates the expected length and is used to load the 3-bit compare register.

A receive interrupt (RX interrupt; see the SCIINTVECT0 and SCIINTVECT1 registers), and a receive ready RXRDY flag set in SCIFLR register, as well as a DMA request (RXDMA) could occur after receiving a response if there are no response receive errors for the frame (such as, there is, frame error, and overrun error).

A transmit interrupt (TX interrupt), and a transmit ready flag (TXRDY flag in SCIFLR register), and a DMA request (TXDMA) could occur after transmitting a response.

Figure 27-8 and Figure 27-9 shows the receive and transmit multi-buffer functional block diagram.

Figure 27-8. Receive Buffers

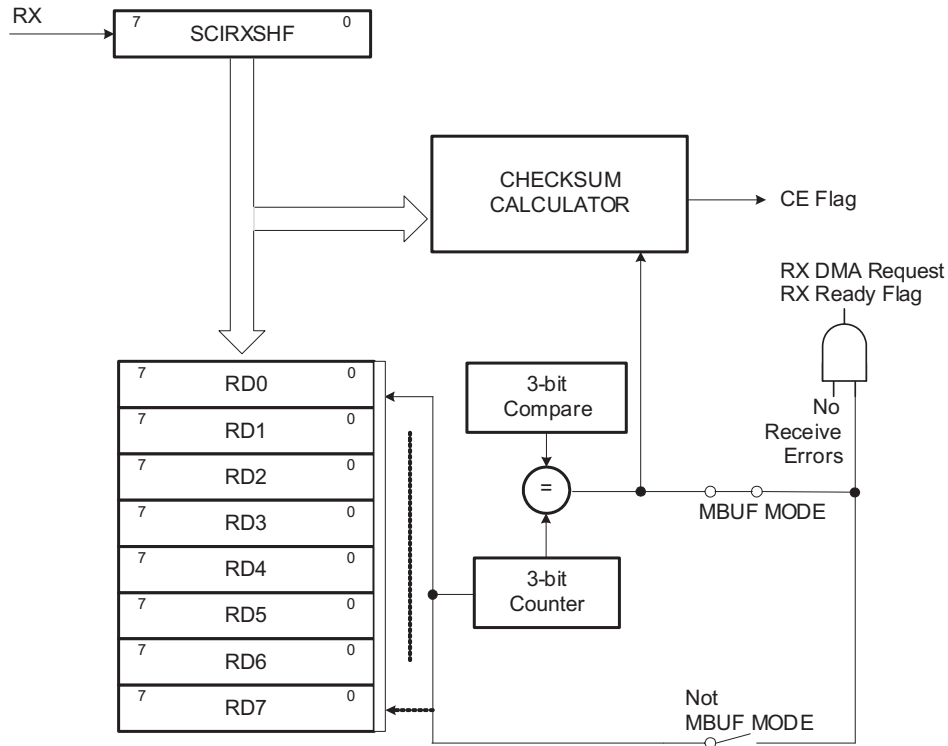
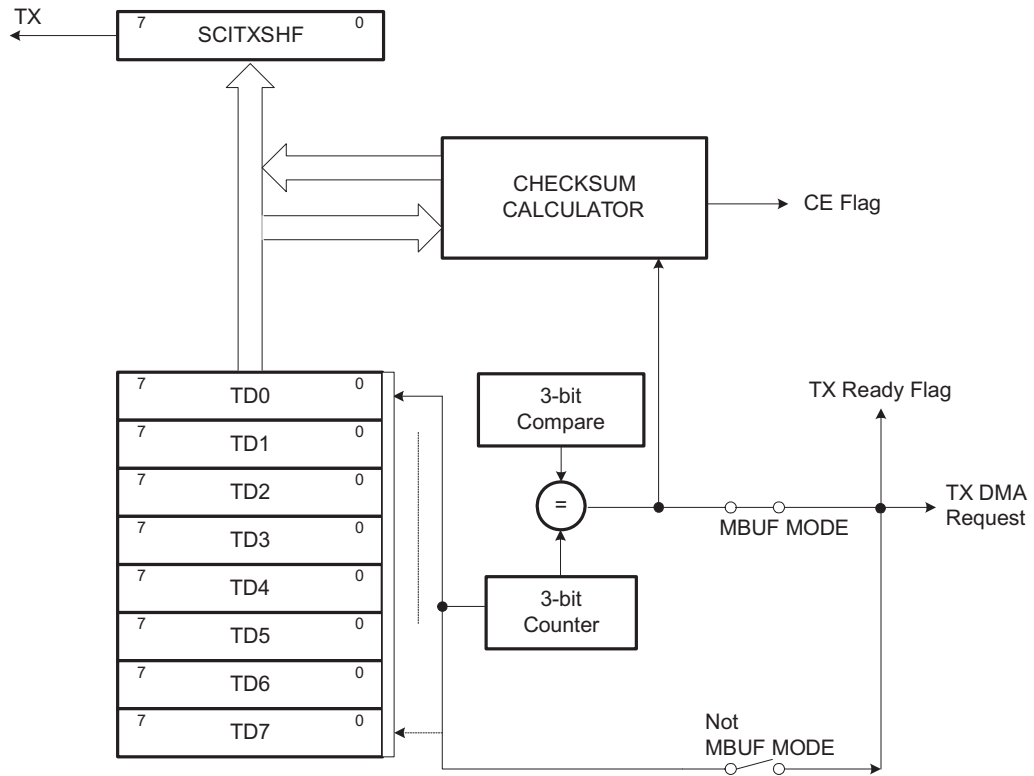


Figure 27-9. Transmit Buffers



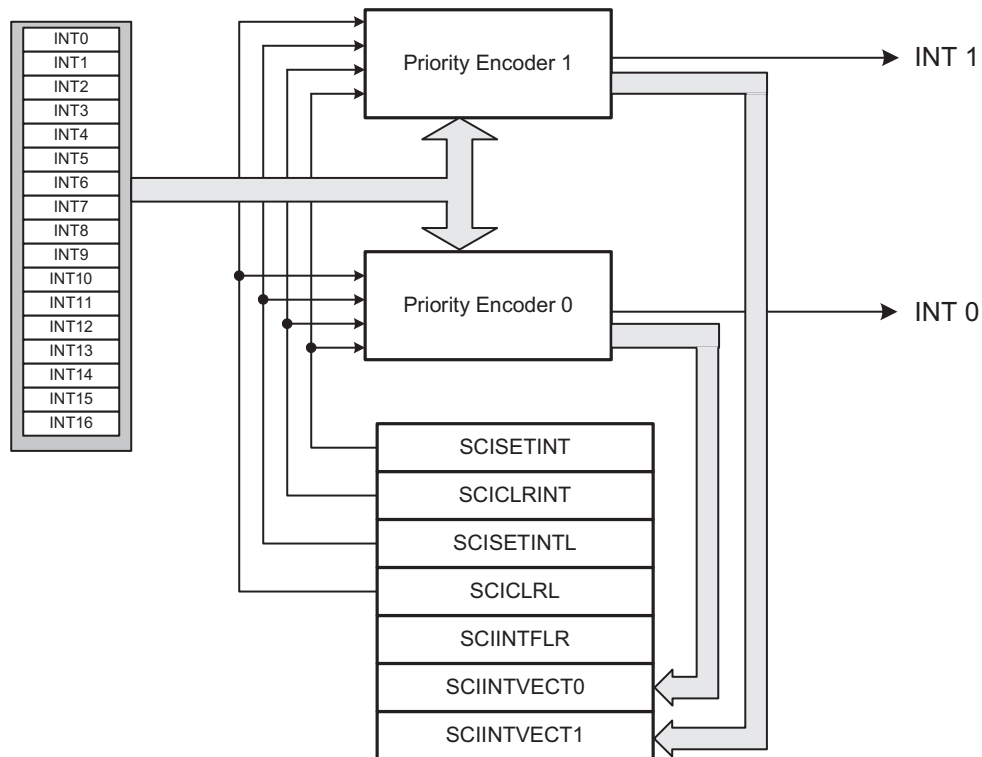
### 27.2.2 SCI Interrupts

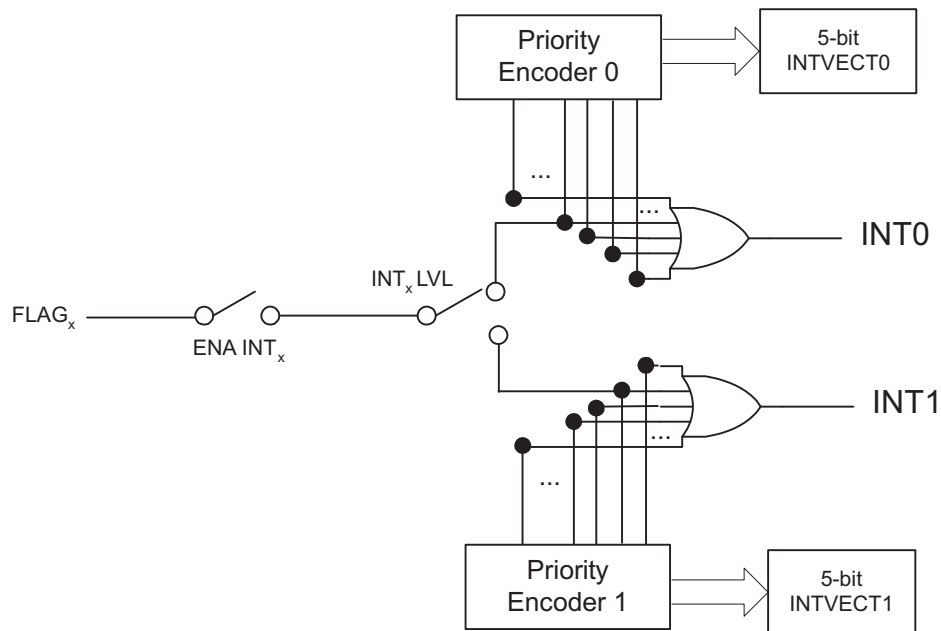
The SCI/LIN module has two interrupt lines, level 0 and level 1, to the vectored interrupt manager (VIM) module (see [Figure 27-10](#)). Two offset registers SCIINTVECT0 and SCIINTVECT1 determine which flag triggered the interrupt according to the respective priority encoders. Each interrupt condition has a bit to enable/disable the interrupt in the SCISSETINT and SCICLRINT registers respectively.

Each interrupt also has a bit that can be set as interrupt level 0 (INT0) or as interrupt level 1 (INT1). By default, interrupts are in interrupt level 0. SCISSETINTLVL sets a given interrupt to level 1. SCICLEARINTLVL resets a given interrupt level to the default level 0.

The interrupt vector registers SCIINTVECT0 and SCIINTVECT1 return the vector of the pending interrupt line INT0 or INT1. If more than one interrupt is pending, the interrupt vector register holds the highest priority interrupt.

**Figure 27-10. General Interrupt Scheme**



**Figure 27-11. Interrupt Generation for Given Flags**


### 27.2.2.1 Transmit Interrupt

To use transmit interrupt functionality, SET TX INT bit must be enabled and SET TX DMA bit must be cleared. The transmit ready (TXRDY) flag is set when the SCI transfers the contents of SCITD to the shift register, SCITXSHF. The TXRDY flag indicates that SCITD is ready to be loaded with more data. In addition, the SCI sets the TX EMPTY bit if both the SCITD and SCITXSHF registers are empty. If the SET TX INT bit is set, then a transmit interrupt is generated when the TXRDY flag goes high. Transmit Interrupt is not generated immediately after setting the SET TX INT bit unlike transmit DMA request. Transmit Interrupt is generated only after the first transfer from SCITD to SCITXSHF, that is first data has to be written to SCITD before any interrupt gets generated. To transmit further data, data can be written to SCITD in the transmit Interrupt service routine.

Writing data to the SCITD register clears the TXRDY bit. When this data has been moved to the SCITXSHF register, the TXRDY bit is set again. The interrupt request can be suspended by setting the CLR TX INT bit; however, when the SET TX INT bit is again set to 1, the TXRDY interrupt is asserted again. The transmit interrupt request can be eliminated until the next series of values is written to SCITD, by disabling the transmitter via the TXENA bit, by a software reset SWnRST, or by a device hardware reset.

### 27.2.2.2 Receive Interrupt

The receive ready (RXRDY) flag is set when the SCI transfers newly received data from SCIRXSHF to SCIRD. The RXRDY flag therefore indicates that the SCI has new data to be read. Receive interrupts are enabled by the SET RX INT bit. If the SET RX INT is set when the SCI sets the RXRDY flag, then a receive interrupt is generated. The received data can be read in the Interrupt Service routine.

On a device with both SCI and a DMA controller, SET RX DMA must be cleared to select interrupt functionality.

### 27.2.2.3 WakeUp Interrupt

SCI sets the WAKEUP flag if bus activity on the RX line either prevents power-down mode from being entered, or RX line activity causes an exit from power-down mode. If enabled (SET WAKEUP INT), wakeup interrupt is triggered once WAKEUP flag is set.



### 27.2.2.4 Error Interrupts

The following error detection are supported with Interrupt by the SCI module:

- Parity errors (PE)
- Frame errors (FE)
- Break Detect errors (BRKDT)
- Overrun errors (OE)
- Bit errors (BE)

If all of these errors (PE, FE, BRKDT, OE, BE) are flagged, an interrupt for the flagged errors will be generated if enabled. A message is valid for both the transmitter and the receiver if there is no error detected until the end of the frame. Each of these flags is located in the receiver status (SCIFLR) register.

There are 16 interrupt sources in the SCI/LIN module, In SCI mode 8 interrupts are supported, as seen in [Table 27-6](#).

**Table 27-6. SCI/LIN Interrupts**

Offset <sup>(1)</sup>	Interrupt	Applicable to SCI	Applicable to LIN
0	No interrupt		
1	Wakeup	Yes	Yes
2	Inconsistent-sync-field error	No	Yes
3	Parity error	Yes	Yes
4	ID	No	Yes
5	Physical bus error	No	Yes
6	Frame error	Yes	Yes
7	Break detect	Yes	No
8	Checksum error	No	Yes
9	Overrun error	Yes	Yes
10	Bit error	Yes	Yes
11	Receive	Yes	Yes
12	Transmit	Yes	Yes
13	No-response error	No	Yes
14	Timeout after wakeup signal (150 ms)	No	Yes
15	Timeout after three wakeup signals (1.5 s)	No	Yes
16	Timeout (Bus Idle, 4s)	No	Yes

<sup>(1)</sup> Offset 1 is the highest priority. Offset 16 is the lowest priority.

**Table 27-7. SCI Receiver Status Flags**

SCI Flag	Register	Bit	Value After Reset <sup>(1)</sup>
CE	SCIFLR	29	0
ISFE	SCIFLR	28	0
NRE	SCIFLR	27	0
FE	SCIFLR	26	0
OE	SCIFLR	25	0
PE	SCIFLR	24	0
RXWAKE	SCIFLR	12	0
RXRDY	SCIFLR	9	0
BUSY	SCIFLR	3	0
IDLE	SCIFLR	2	1
WAKEUP	SCIFLR	1	0
BRKDT	SCIFLR	0	0

<sup>(1)</sup> The flags are frozen with their reset value while SWnRST = 0.

**Table 27-8. SCI Transmitter Status Flags**

SCI Flag	Register	Bit	Value After Reset <sup>(1)</sup>
BE	SCIFLR	31	0
PBE	SCIFLR	30	0
TXWAKE	SCIFLR	10	0
TXEMPTY	SCIFLR	11	1
TXRDY	SCIFLR	8	1

<sup>(1)</sup> The flags are frozen with their reset value while SWnRST = 0.

### 27.2.3 SCI DMA Interface

DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI/LIN module. The DMA transfers depending on whether multi-buffer mode bit (MBUF MODE) is enabled or not enabled. See for more information.

#### 27.2.3.1 Receive DMA Requests

This DMA functionality is enabled/disabled by the CPU using the SET RX DMA/CLR RX DMA bits, respectively.

In Multi-Buffered SCI mode with DMA enabled, the receiver loads the RDy buffers for each received character. RX DMA request is triggered once the last character of the programmed number of characters (LENGTH) are received and copied to the corresponding RDy buffer successfully.

If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis.

In multiprocessor mode, the SCI can generate receiver interrupts for address frames and DMA requests for data frames or DMA requests for both. This is controlled by the SET RX DMA ALL bit.

In multiprocessor mode with the SLEEP bit set, no DMA is generated for received data frames. The software must clear the SLEEP bit before data frames can be received.

#### 27.2.3.2 Transmit DMA Requests

DMA functionality is enabled/disabled by the CPU with SET TX DMA/CLR TX DMA bits, respectively.

In Multi-Buffered SCI mode once TXRDY bit is set or after a transmission of programmed number of characters (LENGTH) (up to eight data bytes stored in the transmit buffer(s) TDy in the LINTD0 and LINTD1 registers), a DMA request is generated in order to reload the transmit buffer for the next transmission. If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis.

## 27.2.4 SCI Configurations

Before the SCI sends or receives data, its registers should be properly configured. Upon power-up or a system-level reset, each bit in the SCI registers is set to a default state. The registers are writable only after the RESET bit in the SCIGCR0 register is set to 1. Of particular importance is the SWnRST bit in the SCIGCR1 register. The SWnRST is an active-low bit initialized to 0 and keeps the SCI in a reset state until it is programmed to 1. Therefore, all SCI configuration should be completed before a 1 is written to the SWnRST bit.

The following list details the configuration steps that software should perform prior to the transmission or reception of data. As long as the SWnRST bit is cleared to 0 the entire time that the SCI is being configured, the order in which the registers are programmed is not important.

- Enable SCI by setting the RESET bit to 1.
- Clear the SWnRST bit to 0 before SCI is configured.
- Select the desired frame format by programming the SCIGCR1 register.
- Set both the RX FUNC and TX FUNC bits in SCPIO0 to 1 to configure the LINRX and LINTX pins for SCI functionality.
- Select the baud rate to be used for communication by programming the BRS register.
- Set the CLOCK bit in SCIGCR1 to 1 to select the internal clock.
- Set the CONT bit in SCIGCR1 to 1 to make SCI not halt for an emulation breakpoint until its current reception or transmission is complete (this bit is used only in an emulation environment).
- Set the LOOP BACK bit in SCIGCR1 to 1 to connect the transmitter to the receiver internally (this feature is used to perform a self-test).
- Set the RXENA bit in SCIGCR1 to 1, if data is to be received.
- Set the TXENA bit in SCIGCR1 to 1, if data is to be transmitted.
- Set the SWnRST bit to 1 after SCI is configured.
- Perform receiving or transmitting data (see [Section 27.2.4.1](#) or [Section 27.2.4.2](#)).

### 27.2.4.1 Receiving Data

The SCI receiver is enabled to receive messages if both the RX FUNC bit and the RXENA bit are set to 1. If the RX FUNC bit is not set, the LINRX pin functions as a general-purpose I/O pin rather than as an SCI function pin.

SCI module can receive data in one of the following modes:

- Single-Buffer (Normal) Mode
- Multi-Buffer Mode

After a valid idle period is detected, data is automatically received as it arrives on the LINRX pin.

#### 27.2.4.1.1 Receiving Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is cleared to 0. In this mode, SCI sets the RXRDY bit when it transfers newly received data from SCIRXSHF to SCIRD. The SCI clears the RXRDY bit after the new data in SCIRD has been read. Also, as data is transferred from SCIRXSHF to SCIRD, the SCI sets the FE, OE, or PE flags if any of these error conditions were detected in the received data. These error conditions are supported with configurable interrupt capability. The wake-up and break-detect status bits are also set if one of these errors occurs, but they do not necessarily occur at the same time that new data is being loaded into SCIRD.

You can receive data by:

1. Polling Receive Ready Flag
2. Receive Interrupt
3. DMA

In polling method, software can poll for the RXRDY bit and read the data from the SCIRD register once the RXRDY bit is set high. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use either the interrupt or DMA method. To use the interrupt method, the SET RX INT bit is set. To use the DMA method, the SET RX DMA bit is set. Either an interrupt or a DMA request is generated the moment the RXRDY bit is set.

#### 27.2.4.1.2 Receiving Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUFMODE bit in SCIGCR1 is set to 1. In this mode, SCI sets the RXRDY bit after receiving the programmed number of data in the receive buffer, the complete frame. The error condition detection logic is similar to the single-buffer mode, except that it monitors for the complete frame. Like single-buffer mode, you can use the polling, DMA, or interrupt method to read the data. The SCI clears the RXRDY bit after the new data in SCIRD has been read.

#### 27.2.4.2 Transmitting Data

The SCI transmitter is enabled if both the TX FUNC bit and the TXENA bit are set to 1. If the TX FUNC bit is not set, the LINTX pin functions as a general-purpose I/O pin rather than as an SCI function pin. Any value written to the SCITD before TXENA is set to 1 is not transmitted. Both of these control bits allow for the SCI transmitter to be held inactive independently of the receiver.

SCI module can transmit data in one of the following modes:

- Single-Buffer (Normal) Mode
- Multi-Buffered or Buffered SCI Mode

##### 27.2.4.2.1 Transmitting Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is cleared to 0. In this mode, SCI waits for data to be written to SCITD, transfers it to SCITXSHF, and transmits the data. The TXRDY and TX EMPTY bits indicate the status of the transmit buffers. That is, when the transmitter is ready for data to be written to SCITD, the TXRDY bit is set. Additionally, if both SCITD and SCITXSHF are empty, then the TX EMPTY bit is also set.

You can transmit data by:

1. Polling Transmit Ready Flag
2. Transmit Interrupt
3. DMA

In polling method, software can poll for the TXRDY bit to go high before writing the data to the SCITD register. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use the interrupt or DMA method. To use the interrupt method, the SET TX INT bit is set. To use the DMA method, the SET TX DMA bit is set. Either an interrupt or a DMA request is generated the moment the TXRDY bit is set. When the SCI has completed transmission of all pending frames, the SCITXSHF register and SCITD are empty, the TXRDY bit is set, and an interrupt/DMA request is generated, if enabled. Because all data has been transmitted, the interrupt/DMA request should be halted. This can either be done by disabling the transmit interrupt (CLR TX INT) or by disabling the transmitter (clear TXENA bit).

---

**NOTE:** The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

---

##### 27.2.4.2.2 Transmitting Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is set to 1. Like single-buffer mode, you can use the polling, DMA, or interrupt method to write the data to be transmitted. The transmitted data has to be written to the SCITD registers. SCI waits for data to be written to the SCITD register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically.

### 27.2.5 SCI Low-Power Mode

The SCI/LIN can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SCI/LIN. During global low-power mode, all clocks to the SCI/LIN are turned off so the module is completely inactive.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI/LIN internal logic and the module registers. Setting the POWERDOWN bit causes the SCI to enter local low-power mode and clearing the POWERDOWN bit causes SCI/LIN to exit from local low-power mode. All the registers are accessible during local power-down mode as any register access enables the clock to SCI for that particular access alone.

The wake-up interrupt is used to allow the SCI to exit low-power mode automatically when a low level is detected on the LINRX pin and also this clears the POWERDOWN bit. If wake-up interrupt is disabled, then the SCI/LIN immediately enters low-power mode whenever it is requested and also any activity on the LINRX pin does not cause the SCI to exit low-power mode.

---

**NOTE: Enabling Local Low-Power Mode During Receive and Transmit**

If the wake-up interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI immediately generates a wake-up interrupt to clear the powerdown bit and prevents the SCI from entering low-power mode and thus completes the current reception. Otherwise, if the wake-up interrupt is disabled, then the SCI completes the current reception and then enters the low-power mode.

---

#### 27.2.5.1 Sleep Mode for Multiprocessor Communication

When the SCI receives data and transfers that data from SCIRXSHF to SCIRD, the RXRDY bit is set and if RX INT ENA is set, the SCI also generates an interrupt. The interrupt triggers the CPU to read the newly received frame before another one is received. In multiprocessor communication modes, this default behavior may be enhanced to provide selective indication of new data. When SCI receives an address frame that does not match its address, the device can ignore the data following this non-matching address until the next address frame by using sleep mode. Sleep mode can be used with both idle-line and address-bit multiprocessor modes.

If sleep mode is enabled by the SLEEP bit, then the SCI transfers data from SCIRXSHF to SCIRD only for address frames. Therefore, in sleep mode, all data frames are assembled in the SCIRXSHF register without being shifted into the SCIRD and without initiating a receive interrupt or DMA request. Upon reception of an address frame, the contents of the SCIRXSHF are moved into SCIRD, and the software must read SCIRD and determine if the SCI is being addressed by comparing the received address against the address previously set in the software and stored somewhere in memory (the SCI does not have hardware available for address comparison). If the SCI is being addressed, the software must clear the SLEEP bit so that the SCI will load SCIRD with the data of the data frames that follow the address frame.

When the SCI has been addressed and sleep mode has been disabled (in software) to allow the receipt of data, the SCI should check the RXWAKE bit (SCIFLR.12) to determine when the next address has been received. This bit is set to 1 if the current value in SCIRD is an address and set to 0 if SCIRD contains data. If the RXWAKE bit is set, then software should check the address in SCIRD against its own address. If it is still being addressed, then sleep mode should remain disabled. Otherwise, the SLEEP bit should be set again.

Following is a sequence of events typical of sleep mode operation:

- The SCI is configured and both sleep mode and receive actions are enabled.
- An address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is not being addressed, so the value of the SLEEP bit is not changed.
- Several data frames are shifted into SCIRXSHF, but no data is moved to SCIRD and no receive interrupts are generated.
- A new address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is being addressed and clears the SLEEP bit.
- Data shifted into SCIRXSHF is transferred to SCIRD, and a receive interrupt is generated after each data frame is received.
- In each interrupt routine, software checks RXWAKE to determine if the current frame is an address frame.
- Another address frame is received, RXWAKE is set, software determines that the SCI is not being addressed and sets the SLEEP bit back to 1. No receive interrupts are generated for the data frames following this address frame.

By ignoring data frames that are not intended for the device, fewer interrupts are generated. These interrupts would otherwise require CPU intervention to read data that is of no significance to this specific device. Using sleep mode can help free some CPU resources.

Except for the RXRDY flag, the SCI continues to update the receiver status flags (see [Table 27-7](#)) while sleep mode is active. In this way, if an error occurs on the receive line, an application can immediately respond to the error and take the appropriate corrective action.

Because the RXRDY bit is not updated for data frames when sleep mode is enabled, the SCI can enable sleep mode and use a polling algorithm if desired. In this case, when RXRDY is set, software knows that a new address has been received. If the SCI is not being addressed, then the software should not change the value of the SLEEP bit and should continue to poll RXRDY.

## 27.3 Local Interconnect Network Module

### 27.3.1 LIN Communication Formats

The SCI/LIN module can be used in LIN mode or SCI mode. The enhancements for baud generation, DMA controls, and additional receive/transmit buffers necessary for LIN mode operation are also part of the enhanced buffered SCI module. LIN mode is selected by enabling LIN MODE bit in SCIGCR1 register.

---

**NOTE:** The SCI/LIN is built around the SCI platform and uses a similar sampling scheme: 16 samples for each bit with majority vote on samples 8, 9, and 10. For the START bit, the first three samples are used.

---

The SCI/LIN control registers are located at the SCI/LIN base address. For a detailed description of each register, see [Section 27.6](#).

#### 27.3.1.1 LIN Standards

For compatibility with LIN2.0 standard the following additional features are implemented over LIN1.3:

- i. Support for LIN 2.0 checksum
- ii. Enhanced synchronizer FSM support for frame processing
- iii. Enhanced handling of extended frames
- iv. Enhanced baud rate generator
- v. Update wakeup/go to sleep

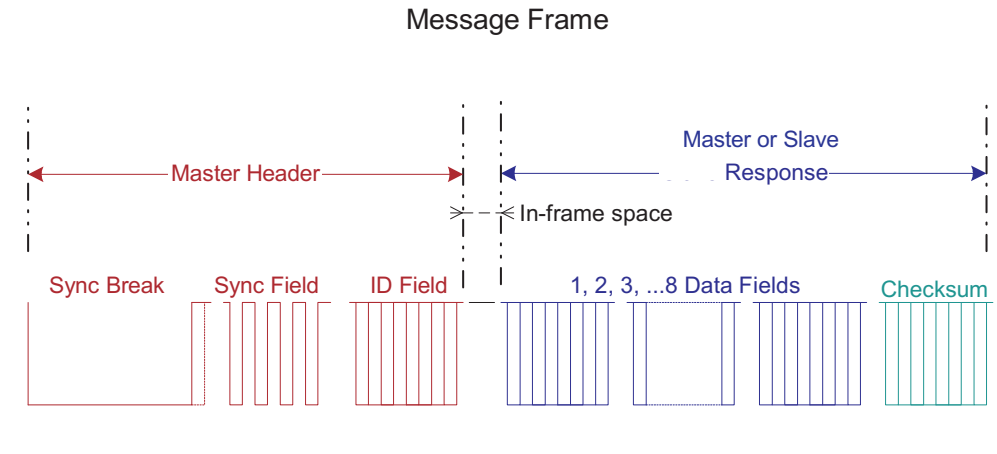
The LIN module covers the CPU performance-consuming features, defined in the *LIN Specification Package Revision 1.3 and 2.0* by hardware.

The Master Mode of LIN module is compatible with LIN 2.1 standard.

### 27.3.1.2 Message Frame

The LIN protocol defines a message frame format, illustrated in [Figure 27-12](#). Each frame includes one master header, one response, one in-frame response space, and inter-byte spaces. In-frame-response and inter-byte spaces may be 0.

**Figure 27-12. LIN Protocol Message Frame Format: Master Header and Slave Response**



There is no arbitration in the definition of the LIN protocol; therefore, multiple slave nodes responding to a header might be detected as an error.

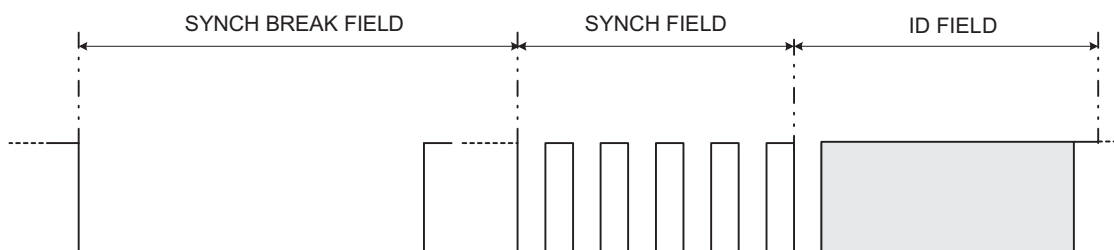
The LIN bus is a single channel wired-AND. The bus has a binary level: either dominant for a value of 0, or recessive for a value of 1.

#### 27.3.1.2.1 Message Header

The header of a message is initiated by a master (see [Figure 27-13](#)) and consists of a three field-sequence:

- The sync break field signaling the beginning of a message
- The sync field conveying bit rate information of the LIN bus
- The ID field denoting the content of a message

**Figure 27-13. Header 3 Fields: Sync Break, Sync, and ID**

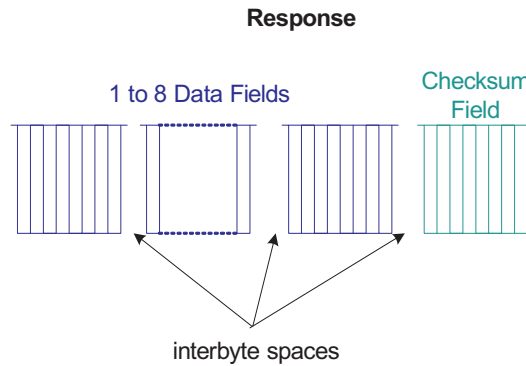




**27.3.1.2.2 Response**

The format of the response is as illustrated in [Figure 27-14](#). There are two types of fields in a response: data and checksum. The data field consists of exactly one data byte, one start bit, and one stop bit, for a total of 10 bits. The LSB is transmitted first. The checksum field consists of one checksum byte, one start bit and one stop bit. The checksum byte is the inverted modulo-256 sum over all data bytes in the data fields of the response.

**Figure 27-14. Response Format of LIN Message Frame**



The format of the response is a stream of N data fields and one checksum field. Typically N is from 1 to 8, with the exception of the extended command frames ([Section 27.3.1.6](#)). The length N of the response is indicated either with the optional length control bits of the ID Field (this is used in standards earlier than LIN 1.x); see [Table 27-9](#), or by LENGTH value in SCIFORMAT[18:16] register; see [Table 27-10](#). The SCI/LIN module supports response lengths from 1 to 8 bytes in compliance with LIN 2.0.

**Table 27-9. Response Length Info Using IDBYTE Field Bits [5:4] for LIN Standards Earlier than 1.3**

ID5	ID4	Number of Data bytes
0	0	2
0	1	2
1	0	4
1	1	8

**Table 27-10. Response Length with SCIFORMAT[18:16] Programming**

SCIFORMAT[18:16]	No. of Bytes
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

### 27.3.1.3 Synchronizer

The synchronizer has three major functions in the messaging between master and slave nodes. It generates the master header data stream, it synchronizes to the LIN bus for responding, and it locally detects timeouts. A bit rate is programmed using the prescalers in the BRSR register to match the indicated LIN\_speed value in the LIN description file.

The LIN synchronizer will perform the following functions: master header signal generation, slave detection and synchronization to message header with optional baud rate adjustment, response transmission timing and timeout control.

The LIN synchronizer is capable of detecting an incoming break and initializing communication at all times.

### 27.3.1.4 Baud Rate

The transmission baud rate of any node is configured by the CPU at the beginning; this defines the bit time  $T_{bit}$ . The bit time is derived from the fields P and M in the baud rate selection register (BRSR). There is an additional 3-bit fractional divider value, field U in the BRSR register, which further fine-tunes the data field baud rate.

The ranges for the prescaler values in the BRSR register are:

$$P = 0, 1, 2, 3, \dots, 2^{24} - 1$$

$$M = 0, 1, 2, \dots, 15$$

$$U = 0, 1, 2, 3, 4, 5, 6, 7$$

The P, M, and U values in the BRSR register are user programmable. The P and M dividers could be used for both SCI mode and LIN mode to select a baud rate. The U value is an additional 3-bit value determining that “ $a T_{VCLK}$ ” (with  $a = 0,1$ ) is added to each  $T_{bit}$  as explained in [Section 27.3.1.4.2](#). If the ADAPT bit is set and the LIN slave is in adaptive baud rate mode, then all these divider values are automatically obtained during header reception when the synchronization field is measured.

The LIN protocol defines baud rate boundaries as follows:

$$1\text{kHz} \leq F_{LINCLK} \leq 20\text{kHz}$$

All transmitted bits are shifted in and out at  $T_{bit}$  periods.

#### 27.3.1.4.1 Fractional Divider

The M field of the BRSR register modifies the integer prescaler P for fine tuning of the baud rate. The M value adds in increments of 1/16 of the P value.

The bit time,  $T_{bit}$  is expressed in terms of the VCLK period  $T_{VCLK}$  as follows:

For all P other than 0, and all M,

$$T_{bit} = 16 \left( P + 1 + \frac{M}{16} \right) T_{VCLK} \quad (16)$$

For P= 0 :  $T_{bit} = 32T_{VCLK}$

Therefore, the LINCLK frequency is given by:

$$F_{LINCLK} = \frac{F_{VCLK}}{16 \left( P + 1 + \frac{M}{16} \right)} \quad \text{For all } P \text{ other than zero}$$

$$F_{LINCLK} = \frac{F_{VCLK}}{32} \quad \text{For } P = 0 \quad (17)$$

### 27.3.1.4.2 Superfractional Divider

The superfractional divider scheme applies to the following modes:

- LIN master mode (sync field + identifier field + response field + checksum field)
- LIN slave mode (response field + checksum field)

#### 27.3.1.4.2.1 Superfractional Divider In LIN Mode

Building on the 4-bit fractional divider M (BRSR[27:24], the superfractional divider uses an additional 3-bit modulating value, illustrated in Table 27-11. The sync field (0x55), the identifier field and the response field can all be seen as 8-bit data bytes flanked by a start bit and a stop bit. The bits with a 1 in the table will have an additional VCLK period added to their  $T_{bit}$ .

**Table 27-11. Superfractional Bit Modulation for LIN Master Mode and Slave Mode <sup>(1)</sup>**

BRSR[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0h	0	0	0	0	0	0	0	0	0	0
1h	1	0	0	0	0	0	0	0	1	0
2h	1	0	0	0	1	0	0	0	1	0
3h	1	0	1	0	1	0	0	0	1	0
4h	1	0	1	0	1	0	1	0	1	0
5h	1	1	1	0	1	0	1	0	1	1
6h	1	1	1	0	1	1	1	0	1	1
7h	1	1	1	1	1	1	1	0	1	1

<sup>(1)</sup>

1. In LIN master mode bit modulation applies to sync field + identifier field + response field
2. In LIN slave mode bit modulation applies to identifier field + response field

The baud rate will vary over a LIN data field to average according to the BRSR[30:28] value by a  $d$  fraction of the peripheral internal clock:  $0 < d < 1$ .

The instantaneous bit time is expressed in terms of  $T_{VCLK}$  as follows:

For all P other than 0, and all M and d (0 or 1),

$$T^{i}bit = \left[ 16 \left( P + 1 + \frac{M}{16} \right) + d \right] T_{VCLK} \quad (18)$$

For P = 0  $T_{bit} = 32T_{VCLK}$

The averaged bit time is expressed in terms of  $T_{VCLK}$  as follows:

For all P other than 0, and all M and d ( $0 < d < 1$ ),

$$T^{a}bit = \left[ 16 \left( P + 1 + \frac{M}{16} \right) + d \right] T_{VCLK} \quad (19)$$

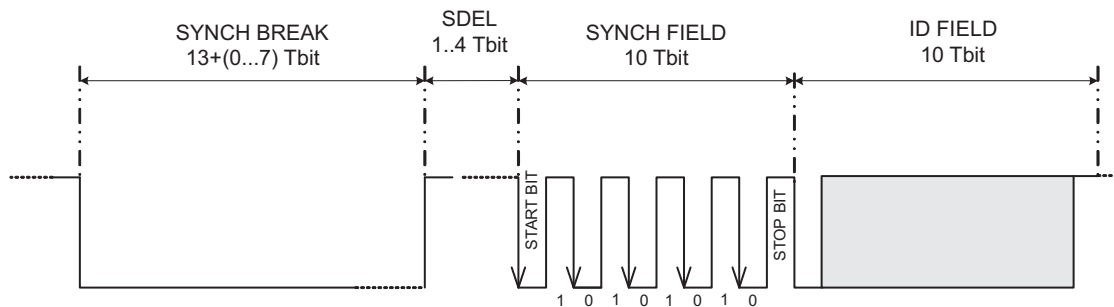
For P = 0  $T_{bit} = 32T_{VCLK}$

### 27.3.1.5 Header Generation

Automatic generation of the LIN protocol header data stream is supported without CPU interaction. The CPU or the DMA will trigger a message header generation and the LIN state machine will handle the generation itself. A master node initiates header generation on CPU or DMA writes to the IDBYTE in the LINID register. The header is always sent by the master to initiate a LIN communication and consists of three fields: break field, synchronization field, and identification field, as seen in [Figure 27-15](#).

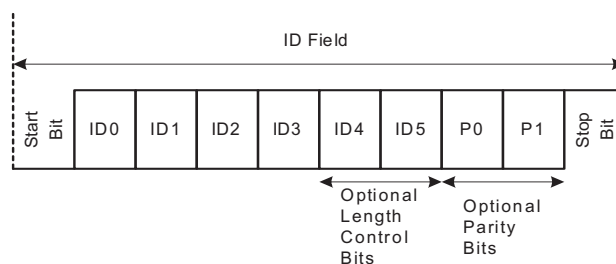
**NOTE:** The LIN protocol uses the parity bits in the identifier. The control length bits are optional to the LIN protocol.

**Figure 27-15. Message Header in Terms of  $T_{bit}$**



- The break field consists of two components:
  - The synchronization break (SYNC BREAK) consists of a minimum of 13 (dominant) low bits to a maximum of 20 dominant bits. The sync break length may be extended from the minimum with the 3-bit SBREAK value in the LINCOMP register.
  - The synchronization break delimiter (SDEL) consists of a minimum of 1 (recessive) high bit to a maximum of 4 recessive bits. The delimiter marks the end of the synchronization break field. The sync break delimiter length depends on the 2-bit SDEL value in the LINCOMP register.
- The synchronization field (SYNC FIELD) consists of one start bit, byte 0x55, and a stop bit. It is used to convey  $T_{bit}$  information and resynchronize LIN bus nodes.
- The identifier field's ID byte may use six bits as an identifier, with optional length control (see *Note: Optional Control Length Bits*), and two optional bits as parity of the identifier. The identifier parity is used and checked if the PARITY\_ENA bit is set. If length control bits are not used, then there can be a total of 64 identifiers plus parity. If neither length control or parity are used there can be up to 256 identifiers. See [Figure 27-16](#) for an illustration of the ID field.

**Figure 27-16. ID Field**



**NOTE: Optional Control Length Bits**

The control length bits only apply to LIN standards prior to LIN 1.3. IDBYTE field conveys response length information if compliant to standards earlier than LIN1.3. The SCIFORMAT register stores the length of the response for later versions of the LIN protocol.

**NOTE:** If the LIN module, configured as Slave in multi-buffer mode, is in the process of transmitting data while a new header comes in, the module might end up in responding with the data from the previous interrupted response (not the data corresponding to the new ID). To avoid this scenario the following procedure could be used:

1. Check for the Bit Error (BE) during the response transmission. If the BE flag is set, this indicates that a collision has happened on the LIN bus (here because of the new Sync Break).
2. In the Bit Error ISR, configure the TD0 and TD1 registers with the next set of data to be transmitted on a TX Match for the incoming ID. Before writing to TD0/TD1 make sure that there was not already an update because of a Bit Error; otherwise TD0/TD1 might be written twice for one ID.
3. Once the complete ID is received, based on the match, the newly configured data will be transmitted by the node.

### 27.3.1.5.1 Event Triggered Frame Handling

The LIN 2.0 protocol uses event-triggered frames that may occasionally cause collisions. Event-triggered frames have to be handled in software.

If no slave answers to an event triggered frame header, the master node will set the NRE flag, and a NRE interrupt will occur if enabled. If a collision occurs, a frame error and checksum error may arise before the NRE error. Those errors are flagged and the appropriate interrupts will occur, if enabled.

Frame errors and checksum errors depend on the behavior and synchronization of the responding slaves. If the slaves are totally synchronized and stop transmission once the collision occurred, it is possible that only the NRE error is flagged despite the occurrence of a collision. To detect if there has been a reception of one byte before the NRE error is flagged, the BUS BUSY flag can be used as an indicator.

The bus busy flag is set on the reception of the first bit of the header and remains set until the header reception is complete, and again is set on the reception of the first bit of the response. In the case of a collision the flag is cleared in the same cycle as the NRE flag is set.

Software could implement the following sequence:

- Once the reception of the header is done (poll for RXID flag), wait for the bus busy flag to get set or NRE flag to get set.
- If bus busy flag is not set before NRE flag, then it is a true no response case (no data has been transmitted onto the bus).
- If bus busy flag gets set, then wait for NRE flag to get set or for successful reception. If NRE flag is set, then in this case a collision has occurred on the bus.

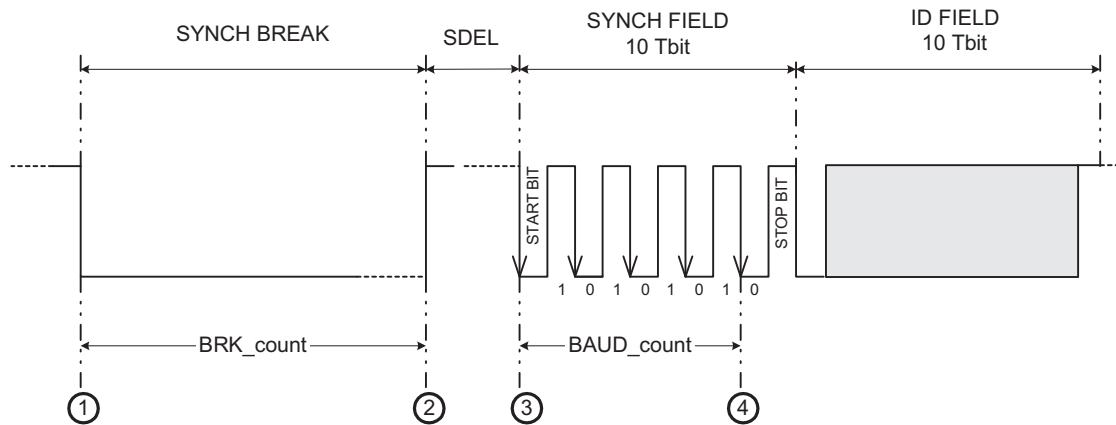
Even in the case of a collision, the received (corrupted) data is accessible in the RX buffers; registers LINRD0 and LINRD1.

### 27.3.1.5.2 Header Reception and Adaptive Baud Rate

A slave node baud rate can optionally be adjusted to the detected bit rate as an option to the LIN module. The adaptive baud rate option is enabled by setting the ADAPT bit. During header reception, a slave measures the baud rate during detection of the sync field. If ADAPT bit is set, then the measured baud rate is compared to the slave node's programmed baud rate and adjusted to the LIN bus baud rate if necessary.

The LIN synchronizer determines two measurements: BRK\_count and BAUD\_count (Figure 27-17). These values are always calculated during the Header reception for sync field validation (Figure 27-18).

**Figure 27-17. Measurements for Synchronization**



By measuring the values BRK\_count and BAUD\_count, a valid sync break sequence can be detected as described in Figure 27-18. The four numbered events in Figure 27-17 signal the start/stop of the synchronizer counter. The synchronizer counter uses VCLK as the time base.

The synchronizer counter is used to measure the sync break relative to the detecting node  $T_{bit}$ . For a slave node receiving the sync break, a threshold of  $11 T_{bit}$  is used as required by the LIN protocol. For detection of the dominant data stream of the sync break, the synchronizer counter is started on a falling edge and stopped on a rising edge of the LINRX. On detection of the sync break delimiter, the synchronizer counter value is saved and then reset.

On detection of five consecutive falling edges, the BAUD\_count is measured. Bit timing calculation and consistency to required accuracy is implemented following the recommendations of LIN revision 2.0. A slave node can calculate a single  $T_{bit}$  time by division of BAUD\_count by 8. In addition, for consistency between the detected edges the following is evaluated:

$$BAUD\_count + BAUD\_count \gg 2 + BAUD\_count \gg 3 \leq BRK\_count$$

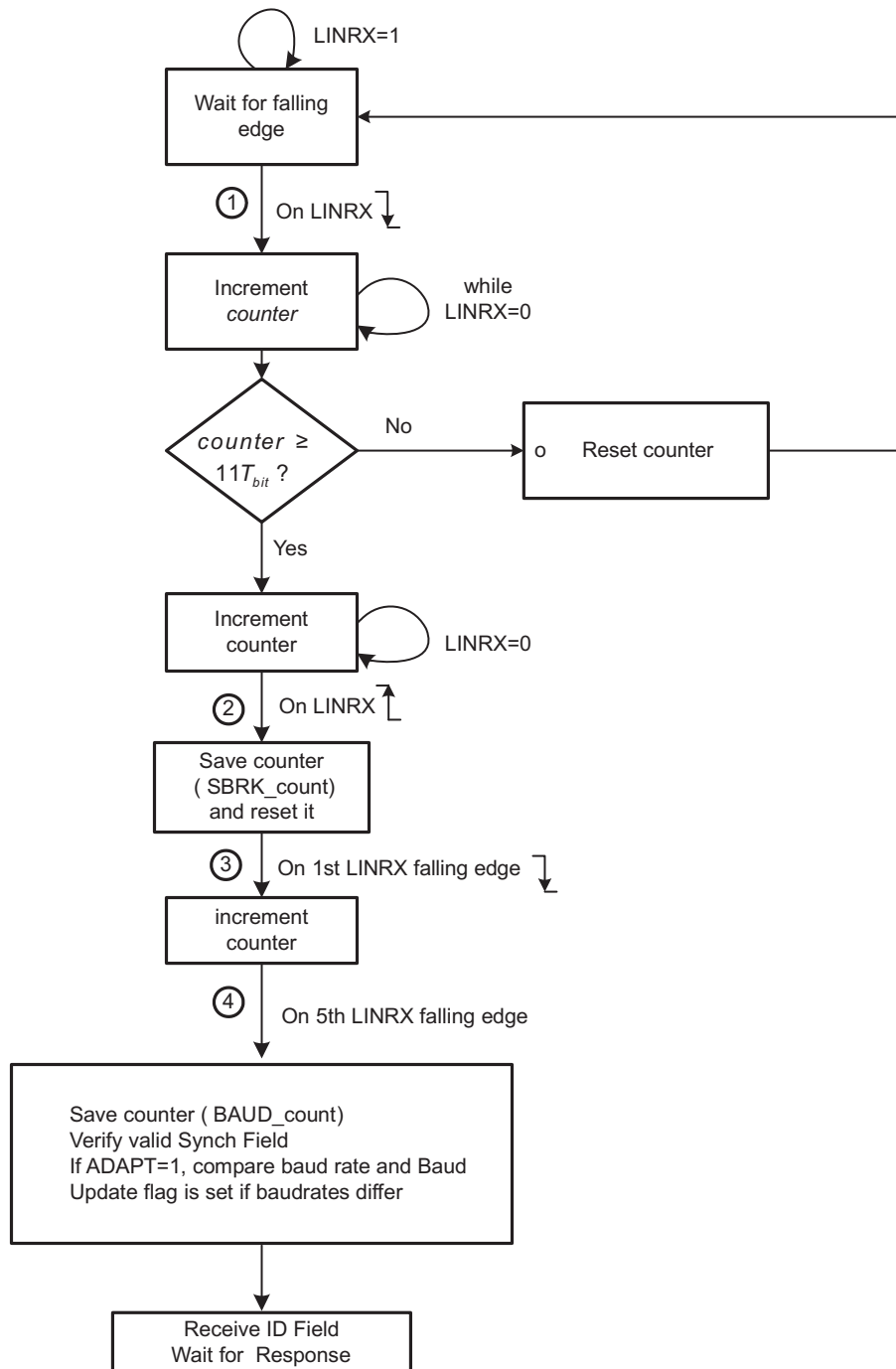
The BAUD\_count value is shifted 3 times to the right and rounded using the first insignificant bit to obtain a  $T_{bit}$  unit. If the ADAPT bit is set, then the detected baud rate is compared to the programmed baud rate.

During the header reception processing as illustrated in Figure 27-18, if the measured BRK\_count value is less than  $11 T_{bit}$ , the sync break is not valid according to the protocol for a fixed rate. If the ADAPT bit is set, then the MBRS register is used for measuring BRK\_count and BAUD\_count values and automatically adjusts to any allowed LIN bus rate (refer to *LIN Specification Package 2.0*).

**NOTE:** In adaptive mode the MBRS divider should be set to allow a maximum baud rate that is not more than 10% above the expected operating baud rate in the LIN network. Otherwise, a 0x00 data byte could mistakenly be detected as a sync break.

The break-threshold relative to the slave node is  $11 T_{bit}$ . The break is  $13 T_{bit}$  as specified in LIN version 1.3.

Figure 27-18. Synchronization Validation Process and Baud Rate Adjustment



If the sync field is not detected within the given tolerances, the inconsistent-sync-field-error (ISFE) flag will be set. An ISFE interrupt will be generated, if enabled by its respective bit in the SCISSETINT register. The ID byte should be received after the sync field validation was successful. Any time a valid break (larger than  $11 T_{bit}$ ) is detected, the receiver's state machine should reset to reception of this new frame. This reset condition is only valid during response state, not if an additional sync break occurs during header reception.

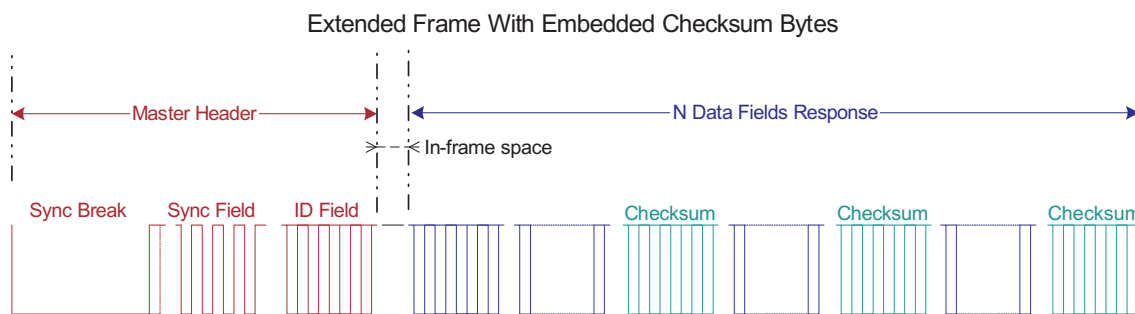
**NOTE:** When an inconsistent sync field (ISFE) error occurs, suggested action for the application is to Reset the SWnRST bit and set the SWnRST bit to make sure that the internal state machines are back to their normal states

### 27.3.1.6 Extended Frames Handling

The LIN protocol 2.0 and prior includes two extended frames with identifiers 62 (user-defined) and 63 (reserved extended). The response data length of the user-defined frame (ID 62, or 0x3E) is unlimited. The length for this identifier will be set at network configuration time to be shared with the LIN bus nodes.

Extended frame communication is triggered on reception of a header with identifier 0x3E; see [Figure 27-19](#). Once the extended frame communication is triggered, unlike normal frames, this communication needs to be stopped before issuing another header. To stop the extended frame communication the STOP EXT FRAME bit must be set.

**Figure 27-19. Optional Embedded Checksum in Response for Extended Frames**



An ID interrupt will be generated (if enabled and there is a match) on reception of ID 62 (0x3E). This interrupt allows the CPU using a software counter to keep track of the bytes that are being sent out and decides when to calculate and insert a checksum byte (recommended at periodic rates). To handle this procedure, SC bit is used. A write to the send checksum bit SC will initiate an automatic send of the checksum byte. The last data field should always be a checksum in compliance with the LIN protocol.

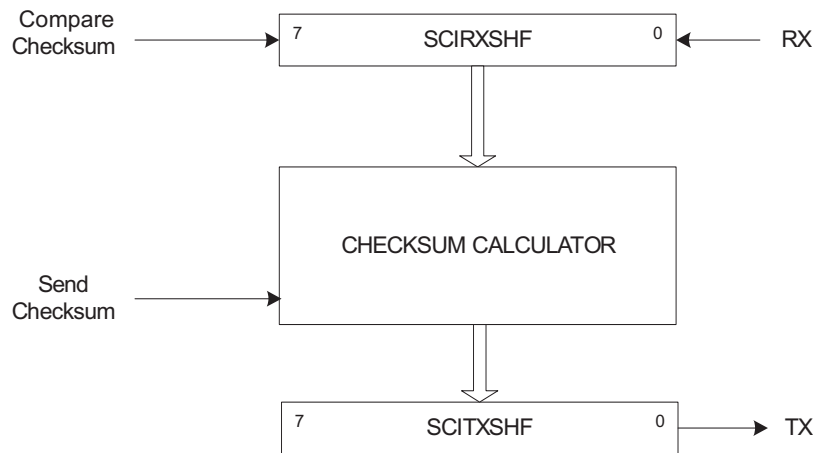
The periodicity of the checksum insertion, defined at network configuration time, is used by the receiving node to evaluate the checksum of the ongoing message, and has the benefit of enhanced reliability.

For the sending node, the checksum is automatically embedded each time the send checksum bit SC is set. For the receiving node, the checksum is compared each time the compare checksum bit CC is set; see [Figure 27-20](#).

**NOTE:** The LIN 2.0 enhanced checksum does not apply to the reserved identifiers. The reserved identifiers always use the classic checksum.



**Figure 27-20. Checksum Compare and Send for Extended Frames**



**27.3.1.7 Timeout Control**

Any LIN node listening to the bus and expecting a response initiated from a master node could flag a no-response error timeout event. The LIN protocol defines four types of timeout events, which are all handled by the hardware of the LIN module. The four LIN protocol events are:

- No-response timeout error
- Bus idle detection
- Timeout after wakeup signal
- Timeout after three wakeup signals

**27.3.1.7.1 No-Response Error (NRE)**

The no-response error will occur when any node expecting a response waits for  $T_{FRAME\_MAX}$  time and the message frame is not fully completed within the maximum length allowed,  $T_{FRAME\_MAX}$ . After this time a no-response error (NRE) is flagged in the NRE bit of the SCIFLR register. An interrupt is triggered if enabled.

As specified in the LIN 1.3 standard, the minimum time to transmit a frame is:

$$T_{FRAME\_MIN} = T_{HEADER\_MIN} + T_{DATA\_FIELD} + T_{CHECKSUM\_FIELD} = 44 + 10N$$

where N = number of data fields.

And the maximum time frame is given by:

$$T_{FRAME\_MAX} = T_{FRAME\_MIN} * 1.4 = (44 + 10N) * 1.4$$

The timeout value  $T_{FRAME\_MAX}$  is derived from the N number of data fields value. The N value is either embedded in the header's ID field for messages or is part of the description file. In the latter case, the 3-bit CHAR value in SCIFORMAT register, will indicate the value for N.

---

**NOTE:** The length coding of the ID field does not apply to two extended frame identifiers, ID fields of 0x3E (62) and 0x3F (63). In these cases, the ID field can be followed by an arbitrary number of data byte fields. Also, the LIN 2.0 protocol specification mentions that ID field 0x3F (63) cannot be used. For these two cases, the NRE will not be handled by the LIN controller hardware.

---

**Table 27-12. Timeout Values in  $T_{bit}$  Units**

N	$T_{DATA\_FIELD}$	$T_{FRAME\_MIN}$	$T_{FRAME\_MAX}$
1	10	54	76
2	20	64	90

**Table 27-12. Timeout Values in  $T_{bit}$  Units (continued)**

<b>N</b>	<b><math>T_{DATA\_FIELD}</math></b>	<b><math>T_{FRAME\_MIN}</math></b>	<b><math>T_{FRAME\_MAX}</math></b>
3	30	74	104
4	40	84	118
5	50	94	132
6	60	104	146
7	70	114	160
8	80	124	174

### 27.3.1.7.2 Bus Idle Detection

The second type of timeout can occur when a node detects an inactive LIN bus: no transitions between recessive and dominant values are detected on the bus. This happens after a minimum of 4 s (this is 80,000  $F_{LINCLK}$  cycles with the fastest bus rate of 20 kbps). If a node detects no activity in the bus as the TIMEOUT bit is set, then it can be assumed that the LIN bus is in sleep mode. Application software can use the Timeout flag to determine when the LIN bus is inactive and put the LIN into sleep mode by writing the POWERDOWN bit.

---

**NOTE:** After the timeout was flagged, a SWnRESET should be asserted before entering Low-Power Mode. This is required to reset the receiver in case that an incomplete frame was on the bus before the idle period.

---

### 27.3.1.7.3 Timeout after Wakeup Signal and Timeout after Three Wakeup Signals

The third and fourth types of timeout are related to the wakeup signal. A node initiating a wakeup should expect a header from the master within a defined amount of time: timeout after wakeup signal. See [Section 27.4.3](#) for more details.

### 27.3.1.8 TXRX Error Detector (TED)

The following sources of error are detected by the TXRX error detector logic (TED). The TED logic consists of a bit monitor, an ID parity checker, and a checksum error. The following errors are detected:

- Bit errors (BE)
- Physical bus errors (PBE)
- Identifier parity errors (PE)
- Checksum errors (CE)

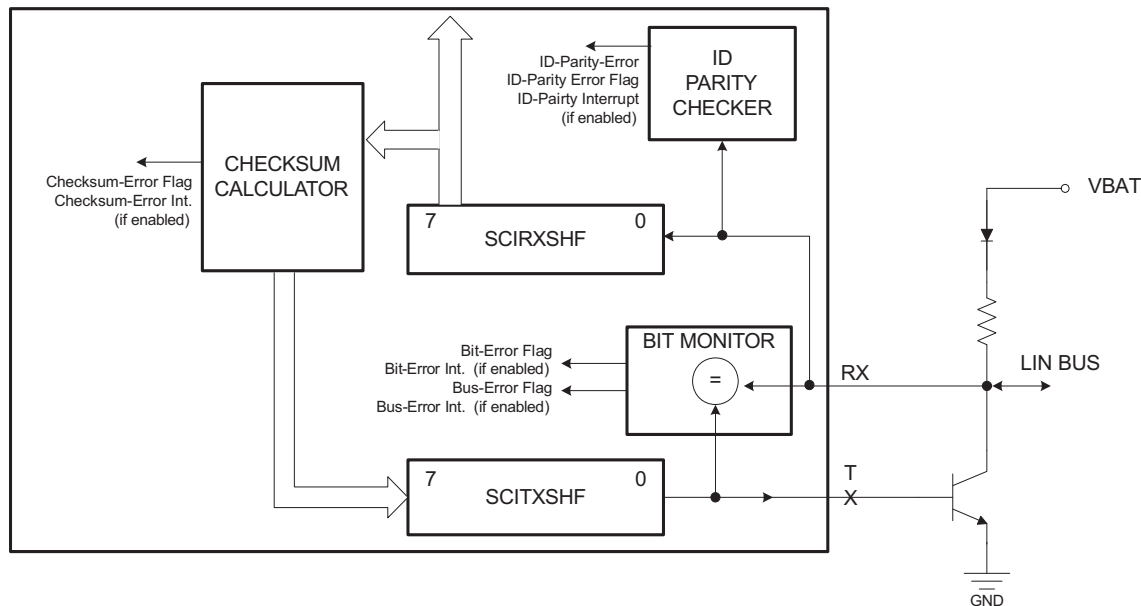
All of these errors (BE, PBE, PE, CE) are flagged. An interrupt for the flagged errors will be generated if enabled. A message is valid for both the transmitter and the receiver if there is no error detected until the end of the frame.

**27.3.1.8.1 Bit Errors**

A bit error (BE) is detected at the bit time when the bit value that is monitored is different from the bit value that is sent. A bit error is indicated by the BE flag in SCIFLR. After signaling a BE, the transmission is aborted no later than the next byte. The bit monitor ensures that the transmitted bit in LINTX is the correct value on the LIN bus by reading back on the LINRX pin as shown in Figure 27-21.

**NOTE:** If a bit occurs due to receiving a header during a slave response, NRE/TIMEOUT flag will not be set for the new frame.

**Figure 27-21. TXRX Error Detector**



**27.3.1.8.2 Physical Bus Errors**

A Physical Bus Error (PBE) has to be detected by a master if no valid message can be generated on the bus (Bus shorted to GND or VBAT). The bit monitor detects a PBE during the header transmission, if no Sync Break can be generated (for example, because of a bus shortage to VBAT) or if no sync break delimiter can be generated (for example, because of a bus shortage to GND). Once the Sync Break Delimiter was validated, all other deviations between the monitored and the sent bit value are flagged as Bit Errors (BE) for this frame.

**27.3.1.8.3 ID Parity Errors**

If parity is enabled, an ID parity error (PE) is detected if any of the two parity bits of the sent ID byte are not equal to the calculated parity on the receiver node. The two parity bits are generated using the following mixed parity algorithm.

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4 \text{ (even Parity)}$$

$$P1 = ID1 \oplus ID3 \oplus ID4 \oplus ID5 \text{ (odd Parity)} \tag{20}$$

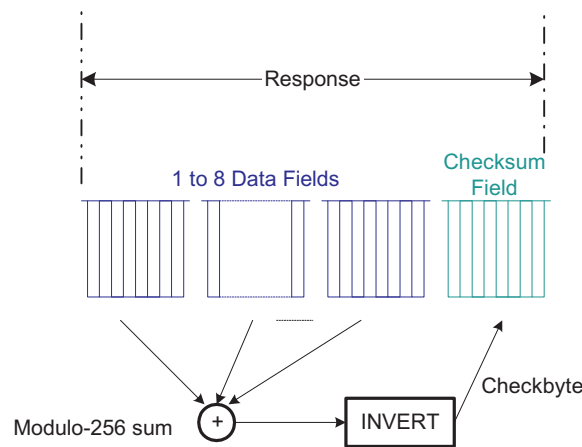
If an ID-parity error is detected, the ID-parity error is flagged, and the received ID is not valid. See Section 27.3.1.9 for details.

### 27.3.1.8.4 Checksum Errors

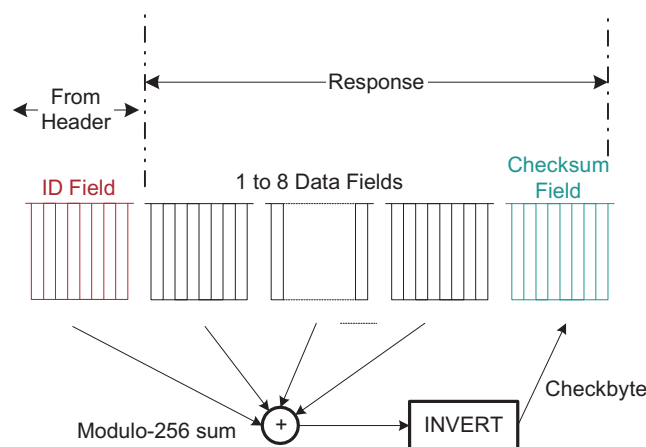
A checksum error (CE) is detected and flagged at the receiving end if the calculated modulo-256 sum over all received data bytes (including the ID byte if it is the enhanced checksum type) plus the checksum byte does not result in 0xFF. The modulo-256 sum is calculated over each byte by adding with carry, where the carry bit of each addition is added to the LSB of its resulting sum.

For the transmitting node, the checksum byte sent at the end of a message is the inverted sum of all the data bytes (see Figure 27-22) for classic checksum implementation. The checksum byte is the inverted sum of the identifier byte and all the data bytes (see Figure 27-23) for the LIN 2.0 compliant enhanced checksum implementation. The classic checksum implementation should always be used for reserved identifiers 60 to 63; therefore, the CTYPE bit will be overridden in this case. For signal-carrying-frame identifiers (0 to 59) the type of checksum used depends on the CTYPE bit.

**Figure 27-22. Classic Checksum Generation at Transmitting Node**



**Figure 27-23. LIN 2.0-Compliant Checksum Generation at Transmitting Node**



### 27.3.1.9 Message Filtering and Validation

Message filtering uses the entire identifier to determine which nodes will participate in a response, either receiving or transmitting a response. Therefore, two acceptance masks are used as shown in [Figure 27-24](#). During header reception, all nodes filter the ID-Field (ID-Field is the part of the header explained in [Figure 27-16](#)) to determine whether they transmit a response or receive a response for the current message. There are two masks for message ID filtering: one to accept a response reception, the other to initiate a response transmission. See [Figure 27-24](#). All nodes compare the received ID to the identifier stored in the ID-SlaveTask BYTE of the LINID register and use the RX ID MASK and the TX ID MASK fields in the LINMASK register to filter the bits of the identifier that should not be compared.

If there is an RX match with no parity error and the RXENA bit is set, there will be an ID RX flag and an interrupt will be triggered if enabled. If there is a TX match with no parity error and the TXENA bit is set, there will be an ID TX flag and an interrupt will be triggered if enabled in the SCISSETINT register.

The masked bits become don't cares for the comparison. To build a mask for a set of identifiers, an XOR function could be used.

For example, to build a mask to accept IDs 0x26 and 0x25 using LINID[7:0] = 0x20; that is, compare 5 most significant bits (MSBs) and filter 3 least significant bits (LSBs), the acceptance mask could be:

$$(0x26 + 0x25) \oplus 0x20 = 0x07 \quad (21)$$

A mask of all zeros will compare all bits of the received identifier in the shift register with the ID-BYTE in LINID[7:0]. If HGEN CTRL is set to 1, a mask of 0xFF will always cause a match. A mask of all 1s will filter all bits of the received identifier, and thus there will be an ID match regardless of the content of the ID-SlaveTask BYTE field in the LINID register.

---

**NOTE:** When the HGEN CTRL bit = 0, the LIN nodes compare the received ID to the ID-BYTE field in the LINID register, and use the RX ID MASK and the TX ID MASK in the LINMASK register to filter the bits of the identifier that should not be compared.

If there is an RX match with no parity error and the RXENA bit is set, there will be an ID RX flag and an interrupt will be triggered if enabled. A mask of all 0s will compare all bits of the received identifier in the shift register with the ID-BYTE field in LINID[7:0]. A mask of all 1s will filter all bits of the received identifier and there will be no match.

---

#### If HGEN CTRL = 1:

- Received ID is compared with the ID-Slave-Task byte, using the RXID mask and the TXID mask.
- A mask of all ones will always result in a match.
- A mask of all zeroes means all the bits must be the same to result in a match.
- If a mask has some bits which are ones, then those bits will not be used for the filtering criterion.

#### If HGEN CTRL = 0:

- Received ID is compared with the ID byte, using the RXID mask and the TXID mask.
- A mask of all ones will result in no match.
- A mask of all zeroes means all the bits must be the same to result in a match.
- If a mask has some bits which are ones, then those bits will not be used for the filtering criterion.

During header reception, the received identifier is copied to the Received ID field LINID[23:16]. If there is no parity error and there is either a TX match or an RX match, then the corresponding TX or RX ID flag is set. If the ID interrupt is enabled, then an ID interrupt is generated.

After the ID interrupt is generated, the CPU may read the Received ID field LINID[23:16] and determine what response to load into the transmit buffers.

---

**NOTE:** When byte 0 is written to TD0 (LINTD0[31:24]), the response transmission is automatically generated.

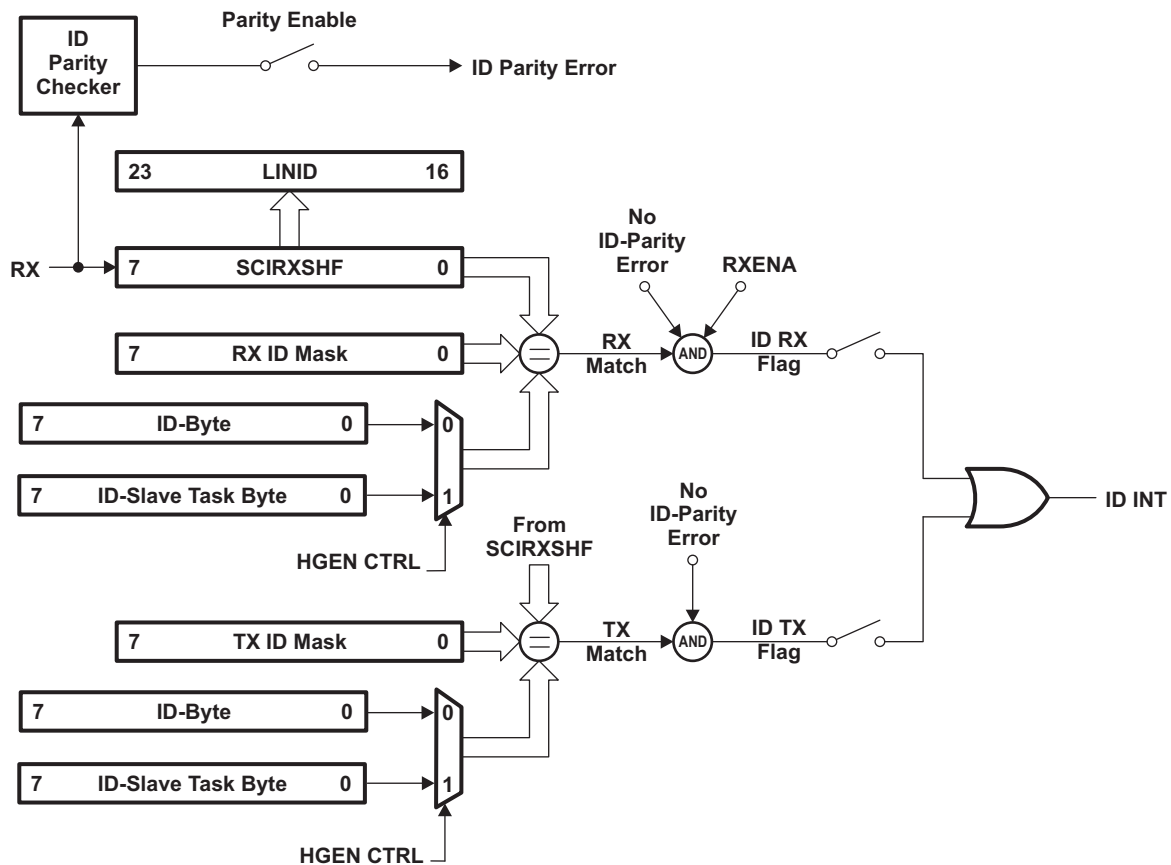
---

In multi-buffer mode, the TXRDY flag will be set when all the response data bytes and checksum byte are copied to the shift register SCITXSHF. In non-multi-buffer mode, the TXRDY flag is set each time a byte is copied to the SCITXSHF register, and also for the last byte of the frame after the checksum byte is copied to the SCITXSHF register.

In multi-buffer mode, the TXEMPTY flag is set when both the transmit buffer(s) TDy and the SCITXSHF shift register are emptied and the checksum has been sent. In nonmulti-buffer mode, TXEMPTY is set each time TD0 and SCITXSHF are emptied, except for the last byte of the frame where the checksum byte must also be transmitted.

If parity is enabled, all slave receiving nodes will validate the identifier using all eight bits of the received ID byte. The SCI/LIN will flag a corrupted identifier if an ID-parity error is detected.

**Figure 27-24. ID Reception, Filtering and Validation**



### 27.3.1.10 Receive Buffers

To reduce CPU load when receiving a LIN N-byte (with  $N = 1-8$ ) response in interrupt mode or DMA mode, the SCI/LIN module has eight receive buffers. These buffers can store an entire LIN response in the RDy receive buffers. [Figure 27-8](#) illustrates the receive buffers.

The checksum byte following the data bytes is validated by the internal checksum calculator. The checksum error (CE) flag indicates a checksum error and a CE interrupt will be generated if enabled in the SCISSETINT register.

The multi-buffer 3-bit counter counts the data bytes transferred from the SCIRXSHF register to the RDy receive buffers if multi-buffer mode is enabled, or to RD0 if multi-buffer mode is disabled. The 3-bit compare register contains the number of data bytes expected to be received. In cases where the ID BYTE field does not convey message length (see *Note: Optional Control Length Bits* in [Section 27.3.1.5](#)), the LENGTH value, indicates the expected length and is used to load the 3-bit compare register. Whether the length control field or the LENGTH value is used is selectable with the COMM MODE bit.

A receive interrupt, and a receive ready RXRDY flag as well as a DMA request (RXDMA) could occur after receiving a response if there are no response receive errors for the frame (such as, there is no checksum error, frame error, and overrun error). The checksum byte will be compared before acknowledging a reception. A DMA request can be generated for each received byte or for the entire response depending on whether the multi-buffer mode is enabled or not (MBUF MODE bit).

---

**NOTE:** In multi-buffer mode following are the scenarios associated with clearing the "RXRDY" flag bit:

1. The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.
  2. For LENGTH less than or equal to 4, Read to RD0 register will clear the "RXRDY" flag.
  3. For LENGTH greater than 4, Read to RD1 register will clear the "RXRDY" flag.
- 

### 27.3.1.11 Transmit Buffers

To reduce the CPU load when transmitting a LIN N-byte (with  $N = 1-8$ ) response in interrupt mode or DMA mode, the SCI/LIN module has eight transmit buffers, TD0–TD7 in LINTD0 and LINTD1. With these transmit buffers, an entire LIN response field can be preloaded in the TXy transmit buffers. Optionally, a DMA transfer could be done on a byte-per-byte basis when multi-buffer mode is not enabled (MBUF MODE bit). [Figure 27-9](#) illustrates the transmit buffers.

The multi-buffer 3-bit counter counts the data bytes transferred from the TDy transmit buffers register if multi-buffer mode is enabled, or from TD0 to SCITXSHF if multi-buffer mode is disabled. The 3-bit compare register contains the number of data bytes expected to be transmitted. If the ID field is not used to convey message length (see *Note: Optional Control Length Bits* in [Section 27.3.1.5](#)), the LENGTH value indicates the expected length and is used instead to load the 3-bit compare register. Whether the length control field or the LENGTH value is used is selectable with the COMM MODE bit.

A transmit interrupt (TX interrupt) and a transmit ready flag (TXRDY flag), as well as a DMA request (TXDMA) could occur after transmitting a response. A DMA request can be generated for each transmitted byte or for the entire response depending on whether multi-buffer mode is enabled or not (MBUF MODE bit).

The checksum byte will be automatically generated by the checksum calculator and sent after the data-fields transmission is finished. The multi-buffer 3-bit counter counts the data bytes transferred from the TDy buffers into the SCITXSHF register.

---

**NOTE:** The transmit interrupt request can be eliminated until the next series of data is written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLRINT register or by disabling the transmitter via the TXENA bit.

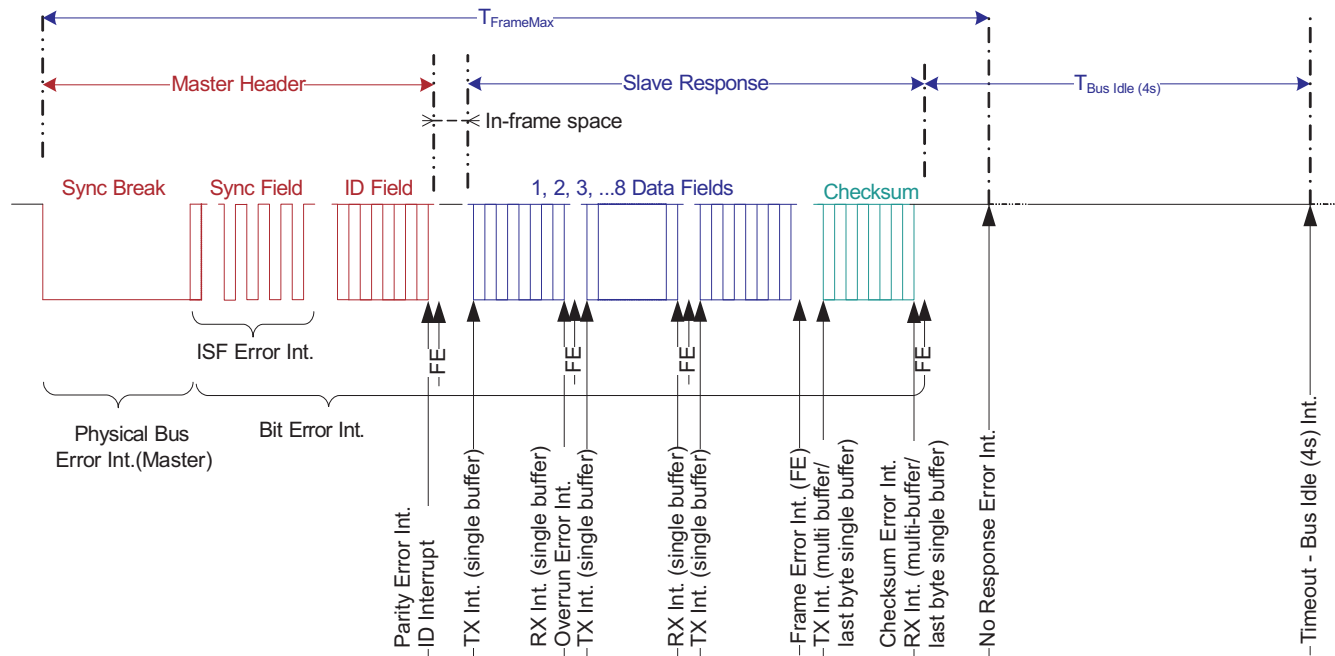
---

### 27.3.2 LIN Interrupts

LIN and SCI mode have a common Interrupt block as explained in Section 27.2.2. There are 16 interrupt sources in the SCI/LIN module, with 8 of them being LIN mode only, as seen in Table 27-6.

A LIN message frame indicating the timing and sequence of the LIN interrupts that could occur is shown in Figure 27-25.

**Figure 27-25. LIN Message Frame Showing LIN Interrupt Timing and Sequence**



### 27.3.3 Servicing LIN interrupts

When servicing an interrupt, clear the corresponding flag in the flag register (SCIFLR) before clearing the global interrupt flag (LIN\_GLB\_INT\_CLR). The ISR should follow the guidelines below. This will prevent any spurious or duplicate interrupt from occurring.

- Clear the LIN interrupt flag in the SCIFLR register.
- Read the LIN interrupt status register to make sure the flag is cleared.
- Clear the global interrupt flag bit in LIN\_GLB\_INT\_CLR.

**NOTE:** The transmit interrupt is generated before the LIN transmitter is ready to accept new data. Inside of the LIN transmit ISR, the software should wait until the buffer is completely empty before loading the next data. This can be done by polling for the Bus Busy Flag (SCIFLR.BUSY) to be 0.

### 27.3.4 LIN DMA Interface

LIN DMA Interface uses the SCI DMA interface logic. DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI/LIN module. There are two modes for DMA transfers depending on whether multi-buffer mode is enabled or not via the multi-buffer enable control bit (MBUF MODE).

**NOTE:** Do not use the DMA to transmit data to multiple slave IDs. Writing to the LINID register will initiate a new transmission. The DMA will write to the LINID register before the LIN state machine is ready to accept the new ID. Doing so will cause the LIN to miss this transmission.



#### 27.3.4.1 LIN Receive DMA Requests

In LIN mode, when the multi-buffer option is enabled, if a received response (up to eight data bytes) is transferred to the receive buffers (RDy), then a DMA request is generated. If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all the expected response data fields are received. This DMA functionality is enabled and disabled using the SET RX DMA and CLR RX DMA bits, respectively.

#### 27.3.4.2 LIN Transmit DMA Requests

In LIN mode with the multi-buffer option enabled, after a transmission (up to eight data bytes stored in the transmit buffer(s) TDy in the LINTD0 and LINTD1 registers), a DMA request is generated in order to reload the transmit buffer for the next transmission. If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all bytes are transferred. This DMA functionality is enabled and disabled using the SET TX DMA and CLR TX DMA bits, respectively.

### 27.3.5 LIN Configurations

The following list details the configuration steps that software should perform prior to the transmission or reception of data in LIN mode. As long as the SWnRST bit in the SCIGCR1 register is cleared to 0 the entire time that the LIN is being configured, the order in which the registers are programmed is not important.

- Enable LIN by setting RESET bit.
- Clear SWnRST to 0 before configuring the LIN.
- Enable the LINRX and LINTX pins by setting the RX FUNC and TX FUNC bits.
- Select LIN mode by programming LIN MODE bit.
- Select Master or Slave mode by programming the CLOCK bit.
- Select the desired frame format (checksum, parity, length control) by programming SCIGCR1.
- Select multi-buffer mode by programming MBUF MODE bit.
- Select the baud rate to be used for communication by programming BRSR.
- Set the maximum baud rate to be used for communication by programming MBRSR.
- Set the CONT bit to make LIN not halt for an emulation breakpoint until its current reception or transmission is complete (this bit is used only in an emulation environment).
- Set LOOP BACK bit to connect the transmitter to the receiver internally if needed (this feature is used to perform a self-test).
- Select the receiver enable RXENA bit if data is to be received.
- Select the transmit enable TXENA bit if data is to be transmitted.
- Select the RX ID MASK and the TX ID MASK fields in the LINMASK register.
- Set SWnRST to 1 after the LIN is configured.
- Perform Receive or Transmit data (see [Section 27.3.1.9](#), [Section 27.3.5.1](#), and [Section 27.3.5.2](#)).

---

**NOTE:** If TXENA is set and the SWnRST is released, the LIN will only generate a new DMA request. The LIN hardware will not generate a new Transmit interrupt request. If using interrupts, the first transmission must be started by software by writing the data to transmit followed by writing to LIN TX to initiate the transmission.

---

#### 27.3.5.1 Receiving Data

The LIN receiver is enabled to receive messages if both the RX FUNC bit and the RXENA bit are set to 1. If the RX FUNC bit is not set, the LINRX pin functions as a general-purpose I/O pin rather than as a LIN function pin.

The ID RX FLAG is set after a valid LIN ID is received with RX Match. An ID interrupt is generated, if enabled.

##### 27.3.5.1.1 Receiving Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit is cleared to 0. In this mode, LIN sets the RXRDY bit when it transfers newly received data from SCIRXSHF to RD0. The SCI clears the RXRDY bit after the new data in RD0 has been read. Also, as data is transferred from SCIRXSHF to RD0, the LIN sets the FE, OE, or PE flags if any of these error conditions were detected in the received data. These error conditions are supported with configurable interrupt capability.

You can receive data by:

1. Polling Receive Ready Flag
2. Receive Interrupt
3. DMA

In polling method, software can poll for the RXRDY bit and read the data from RD0 byte of the LINRD0 register once the RXRDY bit is set high. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use the interrupt or DMA method. To use the interrupt method, the SET RX INT bit is set. To use the DMA method, the SET RX DMA bit must be set. Either an interrupt or a DMA request is generated the moment the RXRDY bit is set. If the checksum scheme is enabled by setting the Compare Checksum (CC) bit to 1, the checksum will be compared on the byte that is currently being received, which is expected to be the checksum byte. The CC bit will be cleared once the checksum is received. A CE will immediately be flagged if there is a checksum error.

### 27.3.5.1.2 Receiving Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit is set to 1. In this mode, LIN sets the RXRDY bit after receiving the programmed number of data in the receive buffer and the checksum field, the complete frame. The error condition detection logic is similar to the single-buffer mode, except that it monitors for the complete frame. Like single-buffer mode, you can use the polling, DMA, or interrupt method to read the data. The received data has to be read from the LINRD0 and LINRD1 registers, based on the number of bytes. For a LENGTH less than or equal to 4, a read from the LINRD0 register clears the RXRDY flag. For a LENGTH greater than 4, a read from the LINRD1 register clears the RXRDY flag. If the checksum scheme is enabled by setting the Compare Checksum (CC) bit to 1 during the reception of the data, then the byte that is received after the reception of the programmed number of data bytes indicated by the LENGTH field is treated as a checksum byte. The CC bit will be cleared once the checksum is received and compared.

### 27.3.5.2 Transmitting Data

The LIN transmitter is enabled if both the TX FUNC bit and the TXENA bit are set to 1. If the TX FUNC bit is not set, the LINTX pin functions as a general-purpose I/O pin rather than as a LIN function pin. Any value written to the TD0 before the TXENA bit is set to 1 is not transmitted. Both of these control bits allow for the LIN transmitter to be held inactive independently of the receiver.

The ID TX flag is set after a valid LIN ID is received with TX Match. An ID interrupt is generated, if enabled.

#### 27.3.5.2.1 Transmitting Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit is cleared to 0. In this mode, LIN waits for data to be written to TD0, transfers it to SCITXSHF, and transmits the data. The TXRDY and TX EMPTY bits indicate the status of the transmit buffers. That is, when the transmitter is ready for data to be written to TD0, the TXRDY bit is set. Additionally, if both TD0 and SCITXSHF are empty, then the TX EMPTY bit is also set.

You can transmit data by:

1. Polling Transmit Ready Flag
2. Transmit Interrupt
3. DMA

In polling method, software can poll for the TXRDY bit to go high before writing the data to the TD0. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use the interrupt or DMA method. To use the interrupt method, the SET TX INT bit is set. To use the DMA method, the SET TX DMA bit is set. Either an interrupt or a DMA request is generated the moment the TXRDY bit is set. When the LIN has completed transmission of all pending frames, the SCITXSHF register and the TD0 are empty, the TXRDY bit is set, and an interrupt/DMA request is generated, if enabled. Because all data has been transmitted, the interrupt/DMA request should be halted. This can either be done by disabling the transmit interrupt (CLR TX INT) / DMA request (CLR TX DMA bit) or by disabling the transmitter (clear TXENA bit). If the checksum scheme is enabled by setting the Send Checksum (SC) bit to 1, the checksum byte will be sent after the current byte transmission. The SC bit will be cleared after the checksum byte has been transmitted.

---

**NOTE:** The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

---

### 27.3.5.2.2 Transmitting Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit is set to 1. Like single-buffer mode, you can use the polling, DMA, or interrupt method to write the data to be transmitted. The transmitted data has to be written to the LINTD0 and LINTD1 registers, based on the number of bytes. LIN waits for data to be written to Byte 0 (TD0) of the LINTD0 register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically. If the checksum scheme is enabled by setting the Send Checksum (SC) bit to 1, the checksum will be sent after transmission of the last byte of the programmed number of data bytes, indicated by the LENGTH field. The SC bit will be cleared after the checksum byte has been transmitted.

## 27.4 Low-Power Mode

The SCI/LIN module can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SCI/LIN module. During global low-power mode, all clocks to the SCI/LIN are turned off so the module is completely inactive. If global low-power mode is requested while the receiver is receiving data, then the SCI/LIN completes the current reception and then enters the low-power mode, that is, module enters low-power mode only when Busy bit (SCIFLR.3) is cleared.

The LIN module may enter low-power mode either when there was no activity on the LINRX pin for more than 4s (this can be either a constant recessive or dominant level) or when a Sleep Command frame was received. Once the Timeout flag (SCIFLR.4) was set or once a Sleep Command was received, the POWERDOWN bit (SCIGCR2.0) must be set by the application software to make the module enter local low-power mode. A wakeup signal will terminate the sleep mode of the LIN bus.

---

**NOTE: Enabling Local Low-Power Mode During Receive and Transmit**

If the wakeup interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI/LIN immediately generates a wake-up interrupt to clear the powerdown bit. Thus, the SCI/LIN is prevented from entering low-power mode and completes the current reception. Otherwise, if the wakeup interrupt is disabled, the SCI/LIN completes the current reception and then enters the low-power mode.

---

### 27.4.1 Entering Sleep Mode

In LIN protocol, a sleep command is used to broadcast the sleep mode to all nodes. The sleep command consists of a diagnostic master request frame with identifier 0x3C (60), with the first data field as 0x00. There should be no activity in the bus once all nodes receive the sleep command: the bus is in sleep mode.

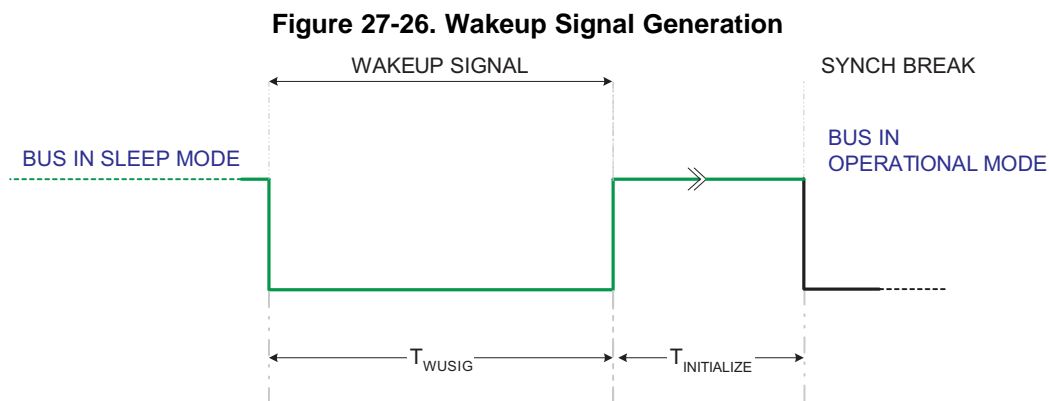
Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI/LIN internal logic and registers. Clearing the POWERDOWN bit causes SCI/LIN to exit from local low-power mode. All the registers are accessible during local power-down mode. If a register is accessed in low-power mode, this access results in enabling the clock to the module for that particular access alone.

### 27.4.2 Wakeup

The wakeup interrupt is used to allow the SCI/LIN module to automatically exit low-power mode. A SCI/LIN wakeup is triggered when a low level is detected on the receive RX pin, and this clears the POWERDOWN bit.

**NOTE:** If the wakeup interrupt is disabled then the SCI/LIN enters low-power mode whenever it is requested to do so, but a low level on the receive RX pin does NOT cause the SCI/LIN to exit low-power mode.

In LIN mode, any node can terminate sleep mode by sending a wakeup signal; see Figure 27-26. A slave node that detects the bus in sleep mode, and with a wakeup request pending, will send a wakeup signal. The wakeup signal is a dominant value on the LIN bus for  $T_{WUSIG}$ ; this is at least  $5 T_{bits}$  for the LIN bus baud rates. The wakeup signal is generated by sending an 0xF0 byte containing 5 dominant  $T_{bits}$  and 5 recessive  $T_{bits}$ .



$$0.25ms \leq T_{WUSIG} \leq 5ms \tag{22}$$

Assuming a perfect bus with no noise or loading effects, a write of 0xF0 to TD0 will load the transmitter to meet the wakeup signal timing requirement for  $T_{WUSIG}$ . Then, setting the GENWU bit will transmit the preloaded value in TD0 for a wakeup signal transmission.

**NOTE:** The GENWU bit can be set/reset only when SWnRST is set to '1' and the node is in power down mode. The bit will be cleared on a valid sync break detection. A master sending a wakeup request, will exit power down mode upon reception of the wakeup pulse. The bit will be cleared on a SWnRST. This can be used to stop a master from sending further wakeup requests.

The TI TPIC1021 LIN transceiver, upon receiving a wakeup signal, will translate it to the microcontroller for wakeup with a dominant level on the RX pin, or a signal to the voltage regulator. While the POWERDOWN bit is set, if the LIN module detects a recessive-to-dominant edge (falling edge) on the RX pin, it will generate a wakeup interrupt if enabled in the SCISSETINT register.

According to LIN protocol 2.0, the TI TPIC1021 LIN transceiver detecting a dominant level on the bus longer than 150 ms will detect it as a wakeup request. The LIN controller's slave is ready to listen to the bus in less than 100 ms ( $T_{INITIALIZE} < 100ms$ ) after a dominant-to-recessive edge (end-of-wakeup signal).

### 27.4.3 Wakeup Timeouts

The LIN protocol defines the following timeouts for a wakeup sequence. After a wakeup signal has been sent to the bus, all nodes wait for the master to send a header. If no sync field is detected before 150 ms (3,000 cycles at 20 kHz) after wakeup signal is transmitted, a new wakeup is sent by the same node that requested the first wakeup. This sequence is not repeated more than two times. After three attempts to wake up the LIN bus, wakeup signal generation is suspended for a 1.5 s (30,000 cycles at 20 kHz) period after three breaks.

---

**NOTE:** To achieve compatibility to LIN1.3 timeout conditions, the MBRS register must be set to assure that the LIN 2.0 (real-time-based) timings meet the LIN 1.3 bit time base. A node triggering the wakeup should set the MBRS register accordingly to meet the targeted time as  $128 \text{ Tbits} \times \text{programmed prescaler}$ .

The LIN controller handles the wakeup expiration times defined by the LIN protocol with a hardware implementation.

---

### 27.5 Emulation Mode

In emulation mode, the CONT bit determines how the SCI/LIN operates when the program is suspended. The SCI/LIN counters are affected by this bit during debug mode. When set, the counters are not stopped and when cleared, the counters are stopped debug mode.

Any reads in emulation mode to a SCI/LIN register will not have any effect on the flags in the SCIFLR register.

---

**NOTE:** When emulation mode is entered during the Frame transmission or reception of the frame and CONT bit is not set, Communication is not expected to be successful. The suggested usage is to set CONT bit during emulation mode for successful communication.

---

## 27.6 SCI/LIN Registers

The SCI/LIN module registers are based on the SCI registers, with added functionality registers enabled by the LIN MODE bit in the SCIGCR1 register.

These registers are accessible in 32-bit reads or writes. The SCI/LIN is controlled and accessed through the registers listed in the following sections. Among the features that can be programmed are the LIN protocol mode, communication and timing modes, baud rate value, frame format, DMA requests, and interrupt configuration.

### 27.6.1 LIN Base Addresses

**Table 27-13. I2C Base Address Table**

Device Registers	Register Name	Start Address	End Address
LinaRegs	LIN_REGS	0000 6A00	0000 6AFF

## 27.6.2 LIN\_REGS Registers

Table 27-14 lists the LIN\_REGS registers. All register offset addresses not listed in Table 27-14 should be considered as reserved locations and the register contents should not be modified.

**Table 27-14. LIN\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	SCIGCR0	Global Control Register 0		<a href="#">Go</a>
4h	SCIGCR1	Global Control Register 1		<a href="#">Go</a>
8h	SCIGCR2	Global Control Register 2		<a href="#">Go</a>
Ch	SCISSETINT	Interrupt Enable Register		<a href="#">Go</a>
10h	SCICLEARINT	Interrupt Disable Register		<a href="#">Go</a>
14h	SCISSETINTLVL	Set Interrupt Level Register		<a href="#">Go</a>
18h	SCICLEARINTLVL	Clear Interrupt Level Register		<a href="#">Go</a>
1Ch	SCIFLR	Flag Register		<a href="#">Go</a>
20h	SCIINTVECT0	Interrupt Vector Offset Register 0		<a href="#">Go</a>
24h	SCIINTVECT1	Interrupt Vector Offset Register 1		<a href="#">Go</a>
28h	SCIFORMAT	Length Control Register		<a href="#">Go</a>
2Ch	BRSR	Baud Rate Selection Register		<a href="#">Go</a>
30h	SCIED	Emulation buffer Register		<a href="#">Go</a>
34h	SCIRD	Receiver data buffer Register		<a href="#">Go</a>
38h	SCITD	Transmit data buffer Register		<a href="#">Go</a>
3Ch	SCIPIO0	Pin control Register 0		<a href="#">Go</a>
44h	SCIPIO2	Pin control Register 2		<a href="#">Go</a>
60h	LINCOMP	Compare register		<a href="#">Go</a>
64h	LINRD0	Receive data register 0		<a href="#">Go</a>
68h	LINRD1	Receive data register 1		<a href="#">Go</a>
6Ch	LINMASK	Acceptance mask register		<a href="#">Go</a>
70h	LINID	LIN ID Register		<a href="#">Go</a>
74h	LINTD0	Transmit Data Register 0		<a href="#">Go</a>
78h	LINTD1	Transmit Data Register 1		<a href="#">Go</a>
7Ch	MBSR	Maximum Baud Rate Selection Register		<a href="#">Go</a>
90h	IODFTCTRL	IODFT for LIN		<a href="#">Go</a>
E0h	LIN_GLB_INT_EN	LIN Global Interrupt Enable Register		<a href="#">Go</a>
E4h	LIN_GLB_INT_FLG	LIN Global Interrupt Flag Register		<a href="#">Go</a>
E8h	LIN_GLB_INT_CLR	LIN Global Interrupt Clear Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 27-15 shows the codes that are used for access types in this section.

**Table 27-15. LIN\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
W1C	W 1C	Write 1 to clear
W1S	W 1S	Write 1 to set
<b>Reset or Default Value</b>		



**Table 27-15. LIN\_REGS Access Type Codes (continued)**

Access Type	Code	Description
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 27.6.2.1 SCIGCR0 Register (Offset = 0h) [reset = 0h]

SCIGCR0 is shown in [Figure 27-27](#) and described in [Table 27-16](#).

Return to the [Summary Table](#).

The SCIGCR0 register defines the module reset.

**Figure 27-27. SCIGCR0 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							RESET
R-0h							R/W-0h

**Table 27-16. SCIGCR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	RESET	R/W	0h	This bit resets the SCI/LIN module. This bit is effective in LIN or SCI-compatible mode.. This bit affects the reset state of the SCI/LIN module. Reset type: SYSRSn 0h (R/W) = SCI/LIN module is in held in reset. 1h (R/W) = SCI/LIN module is out of reset.

### 27.6.2.2 SCIGCR1 Register (Offset = 4h) [reset = 0h]

SCIGCR1 is shown in [Figure 27-28](#) and described in [Table 27-17](#).

Return to the [Summary Table](#).

The SCIGCR1 register defines the frame format, protocol, and communication mode used by the SCI.

**Figure 27-28. SCIGCR1 Register**

31	30	29	28	27	26	25	24
RESERVED						TXENA	RXENA
R-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED						CONT	LOOPBACK
R-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED		STOPEXTFRAME	HGENCTRL	CTYPE	MBUFMODE	ADAPT	SLEEP
R-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
SWnRST	LINMODE	CLK_MASTER	STOP	PARITY	PARITYENA	TIMINGMODE	COMMMODE
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 27-17. SCIGCR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	TXENA	R/W	0h	<p>Transmit enable.</p> <p>This bit is effective in LIN and SCI modes. Data is transferred from SCITD or the TDy (with y=0, 1,...7) buffers in LIN mode to the SCITXSHF shift out register only when the TXENA bit is set.</p> <p>Note: Data written to SCITD or the transmit multi-buffer before TXENA is set is not transmitted. If TXENA is cleared while transmission is ongoing, the data previously written to SCITD is sent (including the checksum byte in LIN mode).</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Disable transfers from SCITD or TDy to SCITXSHF</p> <p>1h (R/W) = Enable transfers of data from SCITD or TDy to SCITXSHF</p>

**Table 27-17. SCIGCR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
24	RXENA	R/W	0h	<p>Receive enable.</p> <p>This bit is effective in LIN or SCI-compatible mode. RXENA allows or prevents the transfer of data from SCIRXSHF to SCIRD or the receive multibuffers.</p> <p>Note: Clearing RXENA stops received characters from being transferred into the receive buffer or multi-buffers, prevents the RX status flags (see Table 7) from being updated by receive data, and inhibits both receive and error interrupts. However, the shift register continues to assemble data regardless of the state of RXENA.</p> <p>Note: If RXENA is cleared before the time the reception of a frame is complete, the data from the frame is not transferred into the receive buffer.</p> <p>Note: If RXENA is set before the time the reception of a frame is complete, the data from the frame is transferred into the receive buffer. If RXENA is set while SCIRXSHF is in the process of assembling a frame, the status flags are not guaranteed to be accurate for that frame. To ensure that the status flags correctly reflect what was detected on the bus during a particular frame, RXENA should be set before the detection of that frame</p> <p>Reset type: SYSRSn                      0h (R/W) = Prevents the receiver from transferring data from the shift buffer to the receive buffer or multi-buffers                      1h (R/W) = Allows the receiver to transfer data from the shift buffer to the receive buffer or multi-buffers</p>
23-18	RESERVED	R	0h	Reserved
17	CONT	R/W	0h	<p>Continue on suspend.</p> <p>This bit has an effect only when a program is being debugged with an emulator, and it determines how the SCI/LIN operates when the program is suspended. This bit affects the LIN counters. When this bit is set, the counters are not stopped during debug. When this bit is cleared, the counters are stopped during debug.</p> <p>Reset type: SYSRSn                      0h (R/W) = When debug mode is entered, the SCI/LIN state machine is frozen. Transmissions and LIN counters are halted and resume when debug mode is exited.                      1h (R/W) = When debug mode is entered, the SCI/LIN continues to operate until the current transmit and receive functions are complete.</p>
16	LOOPBACK	R/W	0h	<p>Loopback bit.</p> <p>This bit is effective in LIN or SCI-compatible mode. The self-checking option for the SCI/LIN can be selected with this bit. If the LINTX and LINRX pins are configured with SCI/LIN functionality, then the LINTX pin is internally connected to the LINRX pin. Externally, during loop back operation, the LINTX pin outputs a high value and the LINRX pin is in a high-impedance state. If this bit value is changed while the SCI/LIN is transmitting or receiving data, errors may result.</p> <p>Reset type: SYSRSn                      0h (R/W) = Loopback mode is disabled.                      1h (R/W) = Loopback mode is enabled.</p>
15-14	RESERVED	R	0h	Reserved

**Table 27-17. SCIGCR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	STOPEXTFRAME	R/W	0h	<p>Stop extended frame communication.</p> <p>This bit is effective in LIN mode only. This bit can be written only during extended frame communication. When the extended frame communication is stopped, this bit is cleared automatically.</p> <p>Reset type: SYSRSn            0h (R/W) = No effect            1h (R/W) = Extended frame communication will be stopped, once current frame transmission/reception is completed.</p>
12	HGENCTRL	R/W	0h	<p>HGEN control bit.</p> <p>This bit is effective in LIN mode only. This bit controls the type of mask filtering comparison.</p> <p>Reset type: SYSRSn            0h (R/W) = ID filtering using ID-Byte.            RECEIVEDID and IDBYTE fields in the LINID register are used for detecting a match (using TX/RXMASK values). Mask of 0xFF in LINMASK register will result in NO match.            1h (R/W) = ID filtering using ID-SlaveTask byte (Recommended).            RECEIVEDID and IDSLAVETASKBYTE fields in the LINID register are used for detecting a match (using TX/RXMASK values). Mask of 0xFF in LINMASK register will result in ALWAYS match</p>
11	CTYPE	R/W	0h	<p>Checksum type.</p> <p>This bit is effective in LIN mode only. This bit controls the type of checksum to be used: classic or enhanced.</p> <p>Reset type: SYSRSn            0h (R/W) = Classic checksum is used.            This checksum is compatible with LIN 1.3 slave nodes. The classic checksum contains the modulo-256 sum with carry over all data bytes. Frames sent with Identifier 60 (0x3C) to 63 (0x3F) must always use the classic checksum.            1h (R/W) = Enhanced checksum is used.            The enhanced checksum is compatible with LIN 2.0 and newer slave nodes. The enhanced checksum contains the modulo-256 sum with carry over all data bytes AND the protected Identifier.</p>
10	MBUFMODE	R/W	0h	<p>Multibuffer mode.</p> <p>This bit is effective in LIN or SCI-compatible mode. This bit controls receive/transmit buffer usage, that is, whether the RX/TX multibuffers are used or a single register, RD0/TD0, is used.</p> <p>Reset type: SYSRSn            0h (R/W) = The multi-buffer mode is disabled.            1h (R/W) = The multi-buffer mode is enabled.</p>
9	ADAPT	R/W	0h	<p>Adapt mode enable.</p> <p>This mode is effective in LIN mode only. This bit has an effect during the detection of the Sync Field. There are two LIN protocol bit rate modes that could be enabled with this bit according to the Node capability file definition: automatic or select. Software and network configuration will decide which of the previous two modes. When this bit is cleared, the LIN 2.0 protocol fixed bit rate should be used. If the ADAPT bit is set, a LIN slave node detecting the baudrate will compare it to the prescalers in BRSR register and update it if they are different. The BRSR register will be updated with the new value. If this bit is not set there will be no adjustment to the BRSR register.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn            0h (R/W) = Automatic baudrate adjustment is disabled.            1h (R/W) = Automatic baudrate adjustment is enabled.</p>

**Table 27-17. SCIGCR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	SLEEP	R/W	0h	<p>SCI sleep.</p> <p>SCI compatibility mode only. In a multiprocessor configuration, this bit controls the receive sleep function. Clearing this bit brings the SCI out of sleep mode.</p> <p>The receiver still operates when the SLEEP bit is set however, RXRDY is updated and SCIRD is loaded with new data only when an address frame is detected. The remaining receiver status flags are updated and an error interrupt is requested if the corresponding interrupt enable bit is set, regardless of the value of the SLEEP bit. In this way, if an error is detected on the receive data line while the SCI is asleep, software can promptly deal with the error condition. The SLEEP bit is not automatically cleared when an address byte is detected.</p> <p>This field is writable in SCI mode only.                      Reset type: SYSRSn                      0h (R/W) = Sleep mode is disabled.                      1h (R/W) = Sleep mode is enabled.</p>
7	SWnRST	R/W	0h	<p>Software reset (active low).</p> <p>This bit is effective in LIN or SCI-compatible mode. The SCI/LIN should only be configured while SWnRST = 0.</p> <p>Only the following configuration bits can be changed in runtime (i.e., while SWnRESET = 1):</p> <ul style="list-style-type: none"> <li>- STOP EXT Frame (SCIGCR1[13])</li> <li>- CC bit (SCIGCR2[17])</li> <li>- SC bit (SCIGCR2[16])</li> </ul> <p>Reset type: SYSRSn                      0h (R/W) = The SCI/LIN is in its reset state no data will be transmitted or received. Writing a 0 to this bit initializes the SCI/LIN state machines and operating flags. All affected logic is held in the reset state until a 1 is written to this bit.                      1h (R/W) = The SCI/LIN is in its ready state transmission and reception can occur. After this bit is set to 1, the configuration of the module should not change.</p>
6	LINMODE	R/W	0h	<p>LIN mode</p> <p>This bit controls the mode of operation of the module.</p> <p>Reset type: SYSRSn                      0h (R/W) = LIN mode is disabled                      SCI compatibility mode is enabled.                      1h (R/W) = LIN mode is enabled                      SCI compatibility mode is disabled.</p>
5	CLK_MASTER	R/W	0h	<p>SCI internal clock enable or LIN Master/Slave configuration.</p> <p>In the SCI mode, this bit enables the clock to the SCI module. In LIN mode, this bit determines whether a LIN node is a slave or master.</p> <p>Reset type: SYSRSn                      0h (R/W) = SCI-compatible mode: Reserved.                      LIN mode: The module is in slave mode.                      1h (R/W) = SCI-compatible mode: Enable clock to the SCI module.                      LIN mode: The node is in master mode.</p>

**Table 27-17. SCIGCR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	STOP	R/W	0h	<p>SCI number of stop bits.</p> <p>This bit is effective in SCI-compatible mode only.</p> <p>Note: The receiver checks for only one stop bit. However in idle-line mode, the receiver waits until the end of the second stop bit (if STOP = 1) to begin checking for an idle period.</p> <p>This field is writable in SCI mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = One stop bit is used.</p> <p>1h (R/W) = Two stop bits are used.</p>
3	PARITY	R/W	0h	<p>SCI parity odd/even selection.</p> <p>This bit is effective in SCI-compatible mode only. If the PARITY ENA bit (SCIGCR1.2) is set, PARITY designates odd or even parity. The parity bit is calculated based on the data bits in each frame and the address bit (in address-bit mode). The start and stop fields in the frame are not included in the parity calculation.</p> <p>This field is writable in SCI mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Odd parity is used. The SCI transmits and expects to receive a value in the parity bit that makes odd the total number of bits in the frame with the value of 1.</p> <p>1h (R/W) = Even parity is used. The SCI transmits and expects to receive a value in the parity bit that makes even the total number of bits in the frame with the value of 1.</p>
2	PARITYENA	R/W	0h	<p>Parity enable.</p> <p>Enables or disables the parity function.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = SCI-compatible mode: Parity disabled no parity bit is generated during transmission or is expected during reception.</p> <p>LIN mode: ID-parity verification is disabled.</p> <p>1h (R/W) = SCI compatible mode: Parity enabled. A parity bit is generated during transmission and is expected during reception.</p> <p>LIN mode: ID-parity verification is enabled.</p>
1	TIMINGMODE	R/W	0h	<p>SCI timing mode bit.</p> <p>This bit is effective in SCI-compatible mode only. It must be set to 1 when the SCI mode is used. This bit configures the SCI for asynchronous operation.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Reserved.</p> <p>1h (R/W) = Must be set to 1 when module is configured for SCI operation</p>
0	COMMMODE	R/W	0h	<p>SCI/LIN communication mode bit.</p> <p>In compatibility mode, it selects the SCI communication mode. In LIN mode it selects length control option for ID-field bits ID4 and ID5.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = SCI-compatible mode: Idle-line mode is used.</p> <p>LIN mode: ID4 and ID5 are not used for length control.</p> <p>1h (R/W) = SCI-compatible mode: Address-bit mode is used.</p> <p>LIN mode: ID4 and ID5 are used for length control.</p>

### 27.6.2.3 SCIGCR2 Register (Offset = 8h) [reset = 0h]

SCIGCR2 is shown in [Figure 27-29](#) and described in [Table 27-18](#).

Return to the [Summary Table](#).

The SCIGCR2 register is used to send or compare a checksum byte during extended frames, to generate a wakeup and for low-power mode control of the LIN module.

**Figure 27-29. SCIGCR2 Register**

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

**Table 27-18. SCIGCR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Reserved
17	CC	R/W	0h	<p>Compare Checksum.</p> <p>This mode is effective in LIN mode only. This bit is used by the receiver for extended frames to trigger a checksum compare. The user will initiate this transaction by writing a one to this bit.</p> <p>In non multibuffer mode, once the CC bit is set, the checksum will be compared on the byte that is currently being received, expected to be the checkbyte.</p> <p>During Multi-buffer mode, following are the scenarios associated with the CC bit :</p> <ul style="list-style-type: none"> <li>- If CC bit is set during the reception of the data, then the byte that is received after the reception of the programmed no. of data bytes indicated by SCIFORMAT[18:16], is treated as a checksum byte.</li> <li>- If CC bit is set during the IDLE period (i.e. during inter-frame space), then the next immediate byte will be treated as a checksum byte.</li> </ul> <p>A CE will immediately be flagged if there is a checksum error. This bit is automatically cleared once the checksum is successfully compared.</p> <p>Reset type: SYSRSn                      0h (R/W) = No effect                      1h (R/W) = Compare checksum on expected checkbyte</p>



**Table 27-18. SCIGCR2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	SC	R/W	0h	<p>Send Checksum</p> <p>This mode is effective in LIN mode only. This bit is used by the transmitter with extended frames to send a checkbyte. In non multibuffer mode the checkbyte will be sent after the current byte transmission. In multibuffer mode the checkbyte will be sent after the last byte count, indicated by the SCIFORMAT[18:16]).</p> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = No checkbyte will be sent. 1h (R/W) = A checkbyte will be sent. This bit will automatically get cleared after the checkbyte is transmitted. The checksum will not be sent if this bit is set before transmitting the very first byte, that is, during interframe space.</p>
15-9	RESERVED	R	0h	Reserved
8	GENWU	R/W	0h	<p>Generate wakeup signal.</p> <p>This bit controls the generation of a wakeup signal, by transmitting the TDO buffer value. This bit is cleared on reception of a valid sync break.</p> <p>Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Transmit TDO for wakeup. This bit will be cleared on a SWnRST (SCIGCR1.7)</p>
7-1	RESERVED	R	0h	Reserved
0	POWERDOWN	R/W	0h	<p>Power down.</p> <p>This bit is effective in LIN or SCI-compatible mode. When the powerdown bit is set, the SCI/LIN module attempts to enter local low-power mode. If the POWERDOWN bit is set while the receiver is actively receiving data and the wakeup interrupt is disabled, then the SCI/LIN will delay low-power mode from being entered until completion of reception. In LIN mode the user may set the POWERDOWN bit on Sleep Command reception or on idle bus detection (more than 4 seconds, i.e. 80,000 cycles at 20kHz)</p> <p>Reset type: SYSRSn 0h (R/W) = Normal operation 1h (R/W) = Request local low-power mode</p>

### 27.6.2.4 SCISSETINT Register (Offset = Ch) [reset = 0h]

SCISSETINT is shown in [Figure 27-30](#) and described in [Table 27-19](#).

Return to the [Summary Table](#).

The SCISSETINT register is used to enable the various interrupts available in the LIN module.

**Figure 27-30. SCISSETINT Register**

31		30		29		28		27		26		25		24	
SETBEINT	SETPBEINT	SETCEINT	SETISFEINT	SETNREINT	SETFEINT	SETOEINT	SETPEINT								
R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h								
23		22		21		20		19		18		17		16	
RESERVED										RESERVED	SET_RX_DMA	SET_TX_DMA			
R-0h										R-0h	R/W1S-0h	R/W1S-0h			
15		14		13		12		11		10		9		8	
RESERVED				SETIDINT	RESERVED				SETRXINT	SETTXINT					
R-0h				R/W1S-0h	R-0h				R/W1S-0h	R/W1S-0h					
7		6		5		4		3		2		1		0	
SETTOA3WUS INT	SETTOAWUSI NT	RESERVED	SETTIMEOUTI NT	RESERVED				SETWAKEUPI NT	SETBRKDTINT						
R/W1S-0h	R/W1S-0h	R-0h	R/W1S-0h	R-0h				R/W1S-0h	R/W1S-0h						

**Table 27-19. SCISSETINT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	SETBEINT	R/W1S	0h	<p>Set bit error interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when there is a bit error.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
30	SETPBEINT	R/W1S	0h	<p>Set physical bus error interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when a physical bus error occurs.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
29	SETCEINT	R/W1S	0h	<p>Set checksum-error Interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when there is a checksum error.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>

**Table 27-19. SCISSETINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28	SETISFEINT	R/W1S	0h	<p>Set inconsistent-sync-field-error interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when there is an inconsistent sync field error.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
27	SETNREINT	R/W1S	0h	<p>Set no-response-error interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when a no-response error occurs.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
26	SETFEINT	R/W1S	0h	<p>Set framing-error interrupt.</p> <p>This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when a framing error occurs.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
25	SETOEINT	R/W1S	0h	<p>Set overrun-error interrupt.</p> <p>This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when an overrun error occurs.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
24	SETPEINT	R/W1S	0h	<p>Set parity interrupt.</p> <p>This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when a parity error occurs.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
23-19	RESERVED	R	0h	Reserved
18	RESERVED	R	0h	Reserved
17	SET_RX_DMA	R/W1S	0h	<p>Set receiver DMA.</p> <p>This bit is effective in LIN or SCI-compatible mode. To enable DMA requests for the receiver this bit must be set. If it is cleared, interrupt requests are generated depending on SETRXINT.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Receiver DMA request is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Receiver DMA request is enabled.</p>

**Table 27-19. SCISSETINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	SET_TX_DMA	R/W1S	0h	<p>Set transmit DMA.</p> <p>This bit is effective in LIN or SCI-compatible mode. To enable DMA requests for the transmitter, this bit must be set. If it is cleared, interrupt requests are generated depending on SETTXINT.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Transmit DMA request is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Transmit DMA request is enabled</p>
15-14	RESERVED	R	0h	Reserved
13	SETIDINT	R/W1S	0h	<p>Set Identification interrupt.</p> <p>This bit is effective in LIN mode only. This bit is set to enable interrupt once a valid matching identifier is received.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
12-10	RESERVED	R	0h	Reserved
9	SETRXINT	R/W1S	0h	<p>Set Receiver interrupt.</p> <p>Setting this bit enables the SCI/LIN to generate a receive interrupt after a frame has been completely received and the data is being transferred from SCIRXSHF to SCIRD.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
8	SETTXINT	R/W1S	0h	<p>Set Transmitter interrupt.</p> <p>Setting this bit enables the SCI/LIN to generate a transmit interrupt as data is being transferred from SCITD to SCITXSHF and the TXRDY bit is being set.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
7	SETTOA3WUSINT	R/W1S	0h	<p>Set Timeout After 3 Wakeup Signals interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN to generate an interrupt when there is a timeout after 3 wakeup signals have been sent.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
6	SETTOAWUSINT	R/W1S	0h	<p>Set Timeout After Wakeup Signal interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN to generate an interrupt when there is a timeout after one wakeup signal has been sent.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
5	RESERVED	R	0h	Reserved

**Table 27-19. SCISSETINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	SETTIMEOUTINT	R/W1S	0h	<p>Set timeout interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN to generate an interrupt when no LIN bus activity (bus idle) occurs for at least 4 seconds.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
3-2	RESERVED	R	0h	Reserved
1	SETWAKEUPINT	R/W1S	0h	<p>Set wake-up interrupt.</p> <p>This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a wake-up interrupt and thereby exit low-power mode. The wake-up interrupt is asserted on falling edge of the wake-up pulse. If enabled, the wake-up interrupt is asserted when local low-power mode is requested while the receiver is busy or if a low level is detected on the SCIRX pin during low-power mode. Wake-up interrupt is not asserted upon a wakeup pulse if the module is not in power down mode.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>
0	SETBRKDTINT	R/W1S	0h	<p>Set break-detect interrupt.</p> <p>This bit is effective in SCI-compatible mode only. Setting this bit enables the SCI/LIN to generate an interrupt if a break condition is detected on the LINRX pin.</p> <p>This field is writable in SCI mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled.</p>

### 27.6.2.5 SCICLEARINT Register (Offset = 10h) [reset = 0h]

SCICLEARINT is shown in [Figure 27-31](#) and described in [Table 27-20](#).

Return to the [Summary Table](#).

The SCICLEARINT register is used to disable the enabled interrupts without accessing the SCISSETINT register.

**Figure 27-31. SCICLEARINT Register**

31		30		29		28		27		26		25		24	
CLRBEINT	CLRPBEINT	CLRCEINT	CLRISFEINT	CLRNREINT	CLRFEINT	CLROEINT	CLRPEINT								
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h								
23		22		21		20		19		18		17		16	
RESERVED										RESERVED		SETRXDMA	CLRTXDMA		
R-0h										R-0h		R/W1C-0h	R/W1C-0h		
15		14		13		12		11		10		9		8	
RESERVED				CLRIDINT	RESERVED				CLRRXINT	CLRTXINT					
R-0h				R/W1C-0h	R-0h				R/W1C-0h	R/W1C-0h					
7		6		5		4		3		2		1		0	
CLRTOA3WUS INT	CLRTOAWUSI NT	RESERVED	CLRTIMEOUTI NT	RESERVED				CLRWAKEUPI NT	CLBRKDTINT						
R/W1C-0h	R/W1C-0h	R-0h	R/W1C-0h	R-0h				R/W1C-0h	R/W1C-0h						

**Table 27-20. SCICLEARINT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	CLRBEINT	R/W1C	0h	<p>Clear Bit Error Interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit disables the bit error interrupt.</p> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect. 1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>
30	CLRPBEINT	R/W1C	0h	<p>Clear Physical Bus Error Interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit disables the physical-bus error interrupt.</p> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect. 1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>
29	CLRCEINT	R/W1C	0h	<p>Clear checksum-error Interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit disables the checksum-error interrupt.</p> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect. 1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>

**Table 27-20. SCICLEARINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28	CLRISFEINT	R/W1C	0h	<p>Clear Inconsistent-Sync-Field-Error Interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit disables the ISFE interrupt.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>
27	CLRNREINT	R/W1C	0h	<p>Clear No-Reponse-Error Interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit disables the no-response error interrupt.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>
26	CLRFEINT	R/W1C	0h	<p>Clear Framing-Error Interrupt.</p> <p>This bit is effective in LIN or SCI-compatible mode. Setting this bit disables framing-error interrupt.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>
25	CLROEINT	R/W1C	0h	<p>Clear Overrun-Error Interrupt.</p> <p>This bit is effective in LIN or SCI-compatible mode. Setting this bit disables the overrun interrupt.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>
24	CLRPEINT	R/W1C	0h	<p>Clear Parity Interrupt.</p> <p>This bit is effective in LIN or SCI-compatible mode. Setting this bit disables the parity error interrupt.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>
23-19	RESERVED	R	0h	Reserved
18	RESERVED	R	0h	Reserved
17	SETRXDMA	R/W1C	0h	<p>Clear receiver DMA.</p> <p>This bit is effective in LIN or SCI-compatible mode. Setting this bit disables the receive DMA request.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Receiver DMA request is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Receiver DMA request is enabled. Writing a 1 to this bit will disable the DMA request and clear this bit.</p>

**Table 27-20. SCICLEARINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	CLRTXDMA	R/W1C	0h	Clear transmit DMA. This bit is effective in LIN or SCI-compatible mode. Setting this bit disables the transmit DMA request. Reset type: SYSRSn 0h (R/W) = Transmit DMA request is disabled. Writing a 0 to this bit has no effect. 1h (R/W) = Transmit DMA request is enabled. Writing a 1 to this bit will disable the DMA request and clear this bit.
15-14	RESERVED	R	0h	Reserved
13	CLRIDINT	R/W1C	0h	Clear Identifier interrupt. This bit is effective in LIN mode only. Setting this bit disables the ID interrupt. Reset type: SYSRSn 0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect. 1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.
12-10	RESERVED	R	0h	Reserved
9	CLRRXINT	R/W1C	0h	Clear Receiver interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit disables the receiver interrupt. Reset type: SYSRSn 0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect. 1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.
8	CLRTXINT	R/W1C	0h	Clear Transmitter interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit disables the transmitter interrupt. Reset type: SYSRSn 0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect. 1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.
7	CLRTOA3WUSINT	R/W1C	0h	Clear Timeout After 3 Wakeup Signals interrupt. This bit is effective in LIN mode only. Setting this bit disables the timeout after 3 wakeup signals interrupt. This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect. 1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.
6	CLRTOAWUSINT	R/W1C	0h	Clear Timeout After Wakeup Signal interrupt. This bit is effective in LIN mode only. Setting this bit disables the timeout after one wakeup signal interrupt. This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect. 1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.
5	RESERVED	R	0h	Reserved



**Table 27-20. SCICLEARINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	CLRTIMEOUTINT	R/W1C	0h	<p>Clear Timeout interrupt.</p> <p>This bit is effective in LIN mode only. Setting this bit disables the timeout (LIN bus idle) interrupt.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>
3-2	RESERVED	R	0h	Reserved
1	CLRWAKEUPINT	R/W1C	0h	<p>Clear Wake-up interrupt.</p> <p>This bit is effective in LIN or SCI-compatible mode. Setting this bit disables the wake-up interrupt.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>
0	CLBRKDTINT	R/W1C	0h	<p>Clear Break-detect interrupt.</p> <p>This bit is effective in SCI-compatible mode only. Setting this bit disables the Break-detect interrupt.</p> <p>This field is writable in SCI mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt is disabled. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt is enabled. Writing a 1 to this bit will disable the interrupt and clear this bit.</p>

**27.6.2.6 SCISSETINTLVL Register (Offset = 14h) [reset = 0h]**

SCISSETINTLVL is shown in [Figure 27-32](#) and described in [Table 27-21](#).

Return to the [Summary Table](#).

The SCISSETINTLVL register is used to map individual interrupt sources to the INT1 interrupt line.

**Figure 27-32. SCISSETINTLVL Register**

31		30		29		28		27		26		25		24	
SETBEINTLVL	SETPBEINTLVL	SETCEINTLVL	SETISFEINTLVL	SETNREINTLVL	SETFEINTLVL	SETOEINTLVL	SETPEINTLVL								
R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h								
23		22		21		20		19		18		17		16	
RESERVED						RESERVED		RESERVED							
R-0h						R-0h		R-0h							
15		14		13		12		11		10		9		8	
RESERVED		SETIDINTLVL		RESERVED						SETRXINTVO		SETTXINTLVL			
R-0h		R/W1S-0h		R-0h						R/W1S-0h		R/W1S-0h			
7		6		5		4		3		2		1		0	
SETTOA3WUSINTLVL	SETTOAWUSINTLVL	RESERVED	SETTIMEOUTINTLVL	RESERVED				SETWAKEUPI	SETBRKDTINTLVL						
R/W1S-0h	R/W1S-0h	R-0h	R/W1S-0h	R-0h				R/W1S-0h	R/W1S-0h						

**Table 27-21. SCISSETINTLVL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	SETBEINTLVL	R/W1S	0h	Set Bit Error interrupt level. This bit is effective in LIN mode only. Writing to this bit maps the Bit Error interrupt level to the INT1 line.  This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt level mapped to INTO line. 1h (R/W) = Interrupt level mapped to INT1 line.
30	SETPBEINTLVL	R/W1S	0h	Set Physical Bus Error interrupt level. This bit is effective in LIN mode only. Writing to this bit maps the Physical Bus Error interrupt level to the INT1 line.  This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt level mapped to INTO line. 1h (R/W) = Interrupt level mapped to INT1 line.
29	SETCEINTLVL	R/W1S	0h	Set Checksum-error interrupt level. This bit is effective in LIN mode only. Writing to this bit maps the Checksum-error interrupt level to the INT1 line.  This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt level mapped to INTO line. 1h (R/W) = Interrupt level mapped to INT1 line.
28	SETISFEINTLVL	R/W1S	0h	Set Inconsistent-Sync-Field-Error interrupt level. This bit is effective in LIN mode only. Writing to this bit maps the Inconsistent-Sync-Field-Error interrupt level to the INT1 line.  This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt level mapped to INTO line. 1h (R/W) = Interrupt level mapped to INT1 line.

**Table 27-21. SCISSETINTLVL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	SETNREINTLVL	R/W1S	0h	<p>Set No-Response-Error interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the No-Response-Error interrupt level to the INT1 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line.</p>
26	SETFEINTLVL	R/W1S	0h	<p>Set Framing-Error interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the Framing-Error interrupt level to the INT1 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line.</p>
25	SETOEINTLVL	R/W1S	0h	<p>Set Overrun-Error Interrupt Level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the Overrun-Error interrupt level to the INT1 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line.</p>
24	SETPEINTLVL	R/W1S	0h	<p>Set Parity Error interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the Parity error interrupt level to the INT1 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line.</p>
23-19	RESERVED	R	0h	Reserved
18	RESERVED	R	0h	Reserved
17-16	RESERVED	R	0h	Reserved
15-14	RESERVED	R	0h	Reserved
13	SETIDINTLVL	R/W1S	0h	<p>Set ID interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the ID interrupt level to the INT1 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line.</p>
12-10	RESERVED	R	0h	Reserved
9	SETRXINTOVO	R/W1S	0h	<p>Set Receiver interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the receiver interrupt level to the INT1 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line.</p>
8	SETTXINTLVL	R/W1S	0h	<p>Set Transmitter interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the transmitter interrupt level to the INT1 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line.</p>

**Table 27-21. SCISSETINTLVL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	SETTOA3WUSINTLVL	R/W1S	0h	<p>Set Timeout After 3 Wakeup Signals interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the timeout after 3 wakeup signals interrupt level to the INT1 line.</p> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt level mapped to INTO line. 1h (R/W) = Interrupt level mapped to INT1 line.</p>
6	SETTOAWUSINTLVL	R/W1S	0h	<p>Set Timeout After Wakeup Signal interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the the timeout after wakeup interrupt level to the INT1 line.</p> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt level mapped to INTO line. 1h (R/W) = Interrupt level mapped to INT1 line.</p>
5	RESERVED	R	0h	Reserved
4	SETTIMEOUTINTLVL	R/W1S	0h	<p>Set Timeout interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the timeout interrupt level to the INT1 line.</p> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = Interrupt level mapped to INTO line. 1h (R/W) = Interrupt level mapped to INT1 line.</p>
3-2	RESERVED	R	0h	Reserved
1	SETWAKEUPINTLVL	R/W1S	0h	<p>Set Wake-up interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the Wake-up interrupt level to the INT1 line. Reset type: SYSRSn 0h (R/W) = Interrupt level mapped to INTO line. 1h (R/W) = Interrupt level mapped to INT1 line.</p>
0	SETBRKDTINTLVL	R/W1S	0h	<p>Set Break-detect interrupt level.</p> <p>This bit is effective in SCI-compatible mode only. Writing to this bit maps the Break-detect interrupt level to the INT1 line.</p> <p>This field is writable in SCI mode only. Reset type: SYSRSn 0h (R/W) = Interrupt level mapped to INTO line. 1h (R/W) = Interrupt level mapped to INT1 line.</p>

### 27.6.2.7 SCICLEARINTLVL Register (Offset = 18h) [reset = 0h]

SCICLEARINTLVL is shown in [Figure 27-33](#) and described in [Table 27-22](#).

Return to the [Summary Table](#).

The SCICLEARINTLVL register is used to map individual interrupt sources to the INT0 line.

**Figure 27-33. SCICLEARINTLVL Register**

31		30		29		28		27		26		25		24		
CLRBEINTLVL	CLRPBEINTLVL	CLRCEINTLVL	CLRISFEINTLVL	CLRNREINTLVL	CLRFEINTLVL	CLROEINTLVL	CLRPEINTLVL									
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h									
23		22		21		20		19		18		17		16		
RESERVED						RESERVED		RESERVED								
R-0h						R-0h		R-0h								
15		14		13		12		11		10		9		8		
RESERVED				CLRIDINTLVL	RESERVED				CLRRXINTLVL	CLRTXINTLVL						
R-0h				R/W1C-0h	R-0h				R/W1C-0h	R/W1C-0h						
7		6		5		4		3		2		1		0		
CLRTOA3WUSINTLVL	CLRTOAWUSINTLVL	RESERVED	CLRTIMEOUTINTLVL	RESERVED				CLRWAKEUPINTLVL	CLRBKDTINTLVL							
R/W1C-0h	R/W1C-0h	R-0h	R/W1C-0h	R-0h				R/W1C-0h	R/W1C-0h							

**Table 27-22. SCICLEARINTLVL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	CLRBEINTLVL	R/W1C	0h	<p>Clear Bit Error interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the Bit Error interrupt level to the INT0 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
30	CLRPBEINTLVL	R/W1C	0h	<p>Clear Physical Bus Error interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the Physical Bus Error interrupt level to the INT0 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
29	CLRCEINTLVL	R/W1C	0h	<p>Clear Checksum-error interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the Checksum-error interrupt level to the INT0 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>

**Table 27-22. SCICLEARINTLVL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28	CLRISFEINTLVL	R/W1C	0h	<p>Clear Inconsistent-Sync-Field-Error interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the Inconsistent-Sync-Field-Error interrupt level to the INT0 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
27	CLRNREINTLVL	R/W1C	0h	<p>Clear No-Response-Error interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the No-Response-Error interrupt level to the INT0 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
26	CLRFEINTLVL	R/W1C	0h	<p>Clear Framing-Error interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the Framing-Error interrupt level to the INT0 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
25	CLROEINTLVL	R/W1C	0h	<p>Clear Overrun-Error Interrupt Level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the Overrun-Error interrupt level to the INT0 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
24	CLRPEINTLVL	R/W1C	0h	<p>Clear Parity Error interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the Parity Error interrupt level to the INT0 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
23-19	RESERVED	R	0h	Reserved
18	RESERVED	R	0h	Reserved
17-16	RESERVED	R	0h	Reserved
15-14	RESERVED	R	0h	Reserved
13	CLRIDINTLVL	R/W1C	0h	<p>Clear ID interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the ID interrupt level to the INT0 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
12-10	RESERVED	R	0h	Reserved

**Table 27-22. SCICLEARINTLVL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	CLRRXINTLVL	R/W1C	0h	<p>Clear Receiver interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the receiver interrupt level to the INT0 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
8	CLRTXINTLVL	R/W1C	0h	<p>Clear Transmitter interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the transmitter interrupt level to the INT0 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
7	CLRTOA3WUSINTLVL	R/W1C	0h	<p>Clear Timeout After 3 Wakeup Signals interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the timeout after 3 wakeup signals interrupt level to the INT0 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
6	CLRTOAWUSINTLVL	R/W1C	0h	<p>Clear Timeout After Wakeup Signal interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the timeout after wakeup interrupt level to the INT0 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
5	RESERVED	R	0h	Reserved
4	CLRTIMEOUTINTLVL	R/W1C	0h	<p>Clear Timeout interrupt level.</p> <p>This bit is effective in LIN mode only. Writing to this bit maps the timeout interrupt level to the INT0 line.</p> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>
3-2	RESERVED	R	0h	Reserved
1	CLRWAKEUPINTLVL	R/W1C	0h	<p>Clear Wake-up interrupt level.</p> <p>This bit is effective in LIN or SCI-compatible mode. Writing to this bit maps the Wake-up interrupt level to the INT0 line.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INT0 line. Writing a 0 to this bit has no effect.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INT0 and clear this bit.</p>

**Table 27-22. SCICLEARINTLVL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	CLRBRKDTINTLVL	R/W1C	0h	<p>Clear Break-detect interrupt level.</p> <p>This bit is effective in SCI-compatible mode only. Writing to this bit maps the Break-detect interrupt level to the INTO line.</p> <p>This field is writable in SCI mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Interrupt level mapped to INTO line.</p> <p>1h (R/W) = Interrupt level mapped to INT1 line. Writing a 1 to this bit will map the interrupt to INTO and clear this bit.</p>



### 27.6.2.8 SCIFLR Register (Offset = 1Ch) [reset = 904h]

SCIFLR is shown in [Figure 27-34](#) and described in [Table 27-23](#).

Return to the [Summary Table](#).

The SCIFLR register indicates the current status of the various interrupt sources of the LIN module.

**Figure 27-34. SCIFLR Register**

31		30		29		28		27		26		25		24	
BE		PBE		CE		ISFE		NRE		FE		OE		PE	
R/W1C-0h		R/W1C-0h		R/W1C-0h		R/W1C-0h		R/W1C-0h		R/W1C-0h		R/W1C-0h		R/W1C-0h	
23		22		21		20		19		18		17		16	
RESERVED															
R-0h															
15		14		13		12		11		10		9		8	
RESERVED		IDRXFLAG		IDTXFLAG		RXWAKE		TXEMPTY		TXWAKE		RXRDY		TXRDY	
R-0h		R/W1C-0h		R/W1C-0h		R-0h		R-1h		R/W-0h		R/W1C-0h		R-1h	
7		6		5		4		3		2		1		0	
TOA3WUS		TOAWUS		RESERVED		TIMEOUT		BUSY		IDLE		WAKEUP		BRKDT	
R/W1C-0h		R/W1C-0h		R-0h		R/W1C-0h		R-0h		R-1h		R/W1C-0h		R/W1C-0h	

**Table 27-23. SCIFLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	BE	R/W1C	0h	<p>Bit Error Flag.</p> <p>This bit is effective in LIN mode only. This bit is set when there has been a bit error. This is detected by the bit monitor in the internal bit monitor. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> <li>- Reception of a new sync break</li> </ul> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No bit error detected.</p> <p>1h (R/W) = Bit error detected.</p>

**Table 27-23. SCIFLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
30	PBE	R/W1C	0h	<p>Physical Bus Error Flag.</p> <p>This bit is effective in LIN mode only. This bit is set when there has been a physical bus error. This is detected by the bit monitor in TED. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> <li>- Reception of a new sync break</li> </ul> <p>Note: this PBE will only be flagged if no sync break can be generated. (because of a bus shortage to VBAT) or if no sync break delimiter can be generated (because of a bus shortage to GND).</p> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = No physical bus error detected. 1h (R/W) = Physical bus error detected.</p>
29	CE	R/W1C	0h	<p>Checksum Error Flag.</p> <p>This bit is effective in LIN mode only. This bit is set when there is checksum error detected by a receiving node. The type of checksum to be used depends on the SCIGCR1.CTYPE bit. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> <li>- Reception of a new sync break</li> </ul> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = No Checksum error detected. 1h (R/W) = Checksum error detected.</p>
28	ISFE	R/W1C	0h	<p>Inconsistent Sync Field Error Flag.</p> <p>This bit is effective in LIN mode only. This bit is set when there has been an inconsistent Sync Field error detected by the synchronizer during header reception. See the "Header Reception and Adaptive Baudrate" section for more information. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> <li>- Reception of a new sync break</li> </ul> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = No Inconsistent Sync Field error detected. 1h (R/W) = Inconsistent Sync Field error detected.</p>

**Table 27-23. SCIFLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	NRE	R/W1C	0h	<p>No-Response Error Flag.</p> <p>This bit is effective in LIN mode only. This bit is set when there is no response to a master's header completed within TFRAME_MAX. This timeout period is applied for message frames of unknown length (identifiers 0 to 61). This error is detected by the synchronizer of the module. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> <li>- Reception of a new sync break</li> </ul> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = No No-Response error detected. 1h (R/W) = No-Response error detected.</p>
26	FE	R/W1C	0h	<p>Framing error flag.</p> <p>This bit is effective in LIN or SCI-compatible mode. This bit is set when an expected stop bit is not found. In SCI compatible mode, only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. Detection of a framing error causes the SCI to generate an error interrupt if the RXERR INT ENA bit is set. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> <li>- Reception of a new character (SCI-compatible mode), or frame (LIN mode)</li> </ul> <p>In multibuffer mode the frame is defined in the SCIFORMAT register. Reset type: SYSRSn 0h (R/W) = No framing error detected. 1h (R/W) = Framing error detected.</p>
25	OE	R/W1C	0h	<p>Overrun error flag.</p> <p>This bit is effective in LIN or SCI-compatible mode. This bit is set when the transfer of data from SCIRXSHF to SCIRD overwrites unread data already in SCIRD or the RDy buffers. Detection of an overrun error causes the LIN to generate an error interrupt if the SET OE INT bit is one. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> </ul> <p>Reset type: SYSRSn 0h (R/W) = No overrun error detected. 1h (R/W) = Overrun error detected.</p>

**Table 27-23. SCIFLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
24	PE	R/W1C	0h	<p>Parity error flag.</p> <p>This bit is effective in LIN or SCI-compatible mode. This bit is set when a parity error is detected in the received data. In SCI address-bit mode, the parity is calculated on the data and address bit fields of the received frame. In idle-line mode, only the data is used to calculate parity. An error is generated when a character is received with a mismatch between the number of 1s and its parity bit. For more information on parity checking, see the "SCI Global Control Register (SCIGCR1)" description. If the parity function is disabled (that is, SCIGCR1.2 = 0), the PE flag is disabled and read as 0. Detection of a parity error causes the LIN to generate an error interrupt if the SET PE INT bit = 1. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Reception of a new character (SCI-compatible mode) or frame (LIN mode)</li> <li>- Writing a 1 to this bit</li> </ul> <p>Reset type: SYSRSn                      0h (R/W) = No parity error or parity disabled.                      1h (R/W) = Parity error detected.</p>
23-16	RESERVED	R	0h	Reserved
15	RESERVED	R	0h	Reserved
14	IDRXFLAG	R/W1C	0h	<p>Identifier On Receive Flag.</p> <p>This bit is effective in LIN mode only. This flag is set once an identifier is received with an RX match and no ID-parity error. See the "Message Filtering and Validation" section for more details. When this flag is set it indicates that a new valid identifier has been received on an RX match. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Reading the LINID register</li> <li>- Writing a 1 to this bit</li> </ul> <p>This field is writable in LIN mode only.                      Reset type: SYSRSn                      0h (R/W) = No valid ID received.                      1h (R/W) = Valid ID RX received in LINID[23:16] on RX match.</p>

**Table 27-23. SCIFLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	IDTXFLAG	R/W1C	0h	<p>Identifier On Transmit Flag.</p> <p>This bit is effective in LIN mode only. This flag is set once an identifier is received with a TX match and no ID-parity error. See the "Message Filtering and Validation" section for more details. When this flag is set it indicates that a new valid identifier has been received on a TX match. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- Setting SWnRESET</li> <li>- System reset</li> <li>- Reading the LINID register</li> <li>- Writing a 1 to this bit</li> </ul> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = No valid ID received. 1h (R/W) = Valid ID received in LINID[23:16] on TX match.</p>
12	RXWAKE	R	0h	<p>Receiver wakeup detect flag.</p> <p>This bit is effective in SCI-compatible mode only. The SCI sets this bit to indicate that the data currently in SCIRD is an address. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- RESET bit</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- System reset</li> <li>- Receipt of a data frame</li> </ul> <p>This bit is writable in SCI mode only. Reset type: SYSRSn 0h (R/W) = The data in SCIRD is not an address. 1h (R/W) = The data in SCIRD is an address. See [1] Section 3.4.4, Sleep Mode for Multiprocessor Communication, on page 16 for more information on using the RXWAKE bit with sleep mode.</p>
11	TXEMPTY	R	1h	<p>Transmitter Empty flag.</p> <p>The value of this flag indicates the contents of the transmitter's buffer register(s) (SCITD/TDy) and shift register (SCITXSHF). In multibuffer mode, this flag indicates the value of the TDx registers and shift register (SCITXSHF). In non multibuffer mode, this flag indicates the value of LINTD0 (byte) and shift register (SCITXSHF). This bit is set by:</p> <ul style="list-style-type: none"> <li>- RESET bit (SCIGCR0.0)</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- System reset.</li> </ul> <p>Note: This bit does not cause an interrupt request. Reset type: SYSRSn 0h (R/W) = Compatible mode or LIN with no multibuffer: Transmitter buffer or shift register (or both) are loaded with data. In LIN mode using multibuffer mode: Multibuffer or shift register (or all) are loaded with data. 1h (R/W) = Compatible mode or LIN with no multibuffer: Transmitter buffer and shift registers are both empty. In LIN mode using multibuffer mode: Multibuffer and shift registers are all empty.</p>

**Table 27-23. SCIFLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	TXWAKE	R/W	0h	<p>SCI transmitter wakeup method select.</p> <p>This bit is effective in SCI-compatible mode only. The TXWAKE bit controls whether the data in SCITD should be sent as an address or data frame using multiprocessor communication format. This bit is set to 1 or 0 by software before a byte is written to SCITD and is cleared by the SCI when data is transferred from SCITD to SCITXSHF or by a system reset. TXWAKE is not cleared by the SWnRESET bit (SCIGCR1.7).</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Address-bit mode: Frame to be transmitted will be data (address bit = 0).</p> <p>Idle-line mode: Frame to be transmitted will be data.</p> <p>1h (R/W) = Address-bit mode: Frame to be transmitted will be an address (address bit=1).</p> <p>Idle-line mode: Following frame to be transmitted will be an address (writing a 1 to this bit followed by writing dummy data to the SCITD will result in a idle period of 11 bit periods before the next frame is transmitted).</p>
9	RXRDY	R/W1C	0h	<p>Receiver ready flag.</p> <p>In SCI compatibility mode, the receiver sets this bit to indicate that the SCIRD contains new data and is ready to be read by the CPU. In LIN mode, RXRDY is set once a valid frame is received in multibuffer mode, a valid frame being a message frame received with no errors. In non multibuffer mode RXRDY is set for each received byte and will be set for the last byte of the frame if there are no errors. The SCI/LIN generates a receive interrupt when RXRDY flag bit is set</p> <p>if the interrupt-enable bit is set (SCISSETINT.9). RXRDY is cleared by:</p> <ul style="list-style-type: none"> <li>- RESET bit (SCIGCR0.0)</li> <li>- Setting the SWnRESET</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> <li>- Reading SCIRD in while in SCI compatibility mode</li> <li>- Reading last data byte RDy of the response in LIN mode</li> </ul> <p>Note: The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVTECT0/1 register.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No new data in SCIRD/RDy.</p> <p>1h (R/W) = New data ready to be read from SCIRD.</p>

**Table 27-23. SCIFLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	TXRDY	R	1h	<p>Transmitter buffer register ready flag.</p> <p>When set, this bit indicates that the transmit buffer(s) register (SCITD in compatibility mode and LINTD0, LINTD1 in MBUF mode) is/are ready to get another character from a CPU write.</p> <p>In SCI compatibility mode, writing data to SCITD automatically clears this bit. In LIN mode, this bit is cleared once byte 0 (TD0) is written to LINTD0. This bit is set after the data of the TX buffer are shifted into the SCITXSHF register. This event can trigger a transmit DMA event if the DMA enable bit is set. This bit is set to 1 by:</p> <ul style="list-style-type: none"> <li>- RESET bit (SCIGCR0.0)</li> <li>- Setting the SWnRESET (SCIGCR1.7)</li> <li>- System reset</li> </ul> <p>Note: The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.</p> <p>Note: The transmit interrupt request can be eliminated until the next series of data is written into the transmit buffers LINTD0 and LINTD1, by disaLing the corresponding interrupt via the SCICLEARINT register or by disaLing the transmitter via the TXENA bit (SCIGCR1.25=0).</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Compatible mode: SCITD is full. LIN mode: The multibuffers are full.</p> <p>1h (R/W) = Compatible mode: SCITD is ready to receive the next character. LIN mode: The multibuffers are ready to receive the next character(s).</p>
7	TOA3WUS	R/W1C	0h	<p>Timeout After 3 Wakeup Signals flag.</p> <p>This bit is effective in LIN mode only. This flag is set if there is no Sync Break received after 3 wakeup signals and a period of 1.5 seconds have passed. Such expiration time is used before issuing another round of wakeup signals. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> </ul> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No timeout after 3 wakeup signals. 1h (R/W) = Timeout after 3 wakeup signals and 1.5s time.</p>

**Table 27-23. SCIFLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	TOAWUS	R/W1C	0h	<p>Timeout After Wakeup Signal flag.</p> <p>This bit is effective in LIN mode only. This bit is set if there is no Sync Break received after a wakeup signal has been sent. A minimum of 150 ms expiration time is used before issuing another wakeup signal. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> </ul> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = No timeout after one wakeup signal (150 ms). 1h (R/W) = Timeout after one wakeup signal.</p>
5	RESERVED	R	0h	Reserved
4	TIMEOUT	R/W1C	0h	<p>LIN Bus IDLE timeout flag.</p> <p>This bit is effective in LIN mode only. This bit is set if there is no LIN bus activity for at least 4 seconds. LIN bus activity being a transition from recessive to dominant. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit</li> </ul> <p>This field is writable in LIN mode only. Reset type: SYSRSn 0h (R/W) = No bus idle detected. 1h (R/W) = LIN bus idle detected.</p>
3	BUSY	R	0h	<p>Bus BUSY flag.</p> <p>This bit is effective in LIN mode and SCI-compatible mode. This bit indicates whether the receiver is in the process of receiving a frame. As soon as the receiver detects the beginning of a start bit, the BUSY bit is set to 1. When the reception of a frame is complete, the BUSY bit is cleared. If SET WAKEUP INT is set and power down is requested while this bit is set, the SCI/LIN automatically prevents low-power mode from being entered and generates wakeup interrupt. The BUSY bit is controlled directly by the SCI receiver but can be cleared by:</p> <ul style="list-style-type: none"> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset.</li> </ul> <p>Reset type: SYSRSn 0h (R/W) = Receiver is not currently receiving a frame. 1h (R/W) = Receiver is currently receiving a frame.</p>



**Table 27-23. SCIFLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	IDLE	R	1h	<p>SCI receiver in idle state.</p> <p>This bit is effective in SCI-compatible mode only. While this bit is set, the SCI looks for an idle period to resynchronize itself with the bit stream. The receiver does not receive any data while the bit is set. The bus must be idle for 11 bit periods to clear this bit. The SCI enters this state:</p> <ul style="list-style-type: none"> <li>- After a system reset</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- After coming out of power down</li> </ul> <p>This bit is writable in SCI mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Idle period detected, the SCI is ready to receive.</p> <p>1h (R/W) = Idle period not detected, the SCI will not receive any data.</p>
1	WAKEUP	R/W1C	0h	<p>Wake-up flag.</p> <p>This bit is effective in LIN mode only. This bit is set by the SCI/LIN when receiver or transmitter activity has taken the module out of power-down mode. An interrupt is generated if the SET WAKEUP INT bit (SCISSETINT.1) is set. This bit is cleared by:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register.</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- Writing a 1 to this bit.</li> </ul> <p>This field is writable in LIN mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Do not wake up from power-down mode.</p> <p>1h (R/W) = Wake up from power-down mode.</p>
0	BRKDT	R/W1C	0h	<p>SCI break-detect flag.</p> <p>This bit is effective in SCI-compatible mode only. This bit is set when the SCI detects a break condition on the LINRX pin. A break condition occurs when the LINRX pin remains continuously low for at least 10 bits after a missing first stop bit, that is, after a framing error. Detection of a break condition causes the SCI to generate an error interrupt if the BRKDT INT ENA bit is set. The BRKDT bit is cleared by the following:</p> <ul style="list-style-type: none"> <li>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register.</li> <li>- Setting the SWnRESET bit (SCIGCR1.7)</li> <li>- RESET bit (SCIGCR0.0)</li> <li>- System reset</li> <li>- By writing a 1 to this bit.</li> </ul> <p>This bit is writable in SCI mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No break condition detected.</p> <p>1h (R/W) = Break condition detected.</p>

### 27.6.2.9 SCIINTVECT0 Register (Offset = 20h) [reset = 0h]

SCIINTVECT0 is shown in [Figure 27-35](#) and described in [Table 27-24](#).

Return to the [Summary Table](#).

The SCIINTVECT0 register indicates the offset for the INT0 interrupt line.

**Figure 27-35. SCIINTVECT0 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											INTVECT0				
R-0h											R-0h				

**Table 27-24. SCIINTVECT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-5	RESERVED	R	0h	Reserved
4-0	INTVECT0	R	0h	Interrupt vector offset for INT0. This register indicates the offset for interrupt line INT0. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag corresponding to the offset that was read.  Note: The flags for the receive (SCIFLR.9) and the transmit (SCIFLR.8) interrupts cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register). Reset type: SYSRSn

**27.6.2.10 SCIINTVECT1 Register (Offset = 24h) [reset = 0h]**

SCIINTVECT1 is shown in [Figure 27-36](#) and described in [Table 27-25](#).

Return to the [Summary Table](#).

The SCIINTVECT1 register indicates the offset for the INT1 interrupt line.

**Figure 27-36. SCIINTVECT1 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											INTVECT1				
R-0h											R-0h				

**Table 27-25. SCIINTVECT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-5	RESERVED	R	0h	Reserved
4-0	INTVECT1	R	0h	<p>Interrupt vector offset for INT1.</p> <p>This register indicates the offset for interrupt line INT1. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag corresponding to the offset that was read.</p> <p>Note: The flags for the receive (SCIFLR.9) and the transmit (SCIFLR.8) interrupts cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register).</p> <p>Reset type: SYSRSn</p>

**27.6.2.11 SCIFORMAT Register (Offset = 28h) [reset = 0h]**

SCIFORMAT is shown in [Figure 27-37](#) and described in [Table 27-26](#).

Return to the [Summary Table](#).

The SCIFORMAT register is used to set up the character and frame lengths.

**Figure 27-37. SCIFORMAT Register**

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

**Table 27-26. SCIFORMAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18-16	LENGTH	R/W	0h	<p>Frame length control bits.</p> <p>In LIN mode, these bits indicate the number of bytes in the response field from 1 to 8 bytes. In buffered SCI mode, these bits indicate the number of characters. When these bits are used to indicate LIN response length (SCIGCR1[0] = 1), then when there is an ID RX match, this value should be updated with the expected length of the response. In buffered SCI mode, these bits indicate the number of characters with SCIFORMAT[2:0] bits per character. i.e. these bits indicate the transmitter/receiver format for the number of characters: 1 to 8. There can be up to eight characters with eight bits each.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The response field has 1 bytes/characters.                      1h (R/W) = The response field has 2 bytes/characters.                      2h (R/W) = The response field has 3 bytes/characters.                      3h (R/W) = The response field has 4 bytes/characters.                      4h (R/W) = The response field has 5 bytes/characters.                      5h (R/W) = The response field has 6 bytes/characters.                      6h (R/W) = The response field has 7 bytes/characters.                      7h (R/W) = The response field has 8 bytes/characters.</p>
15-3	RESERVED	R	0h	Reserved

**Table 27-26. SCIFORMAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2-0	CHAR	R/W	0h	<p>Character length control bits.</p> <p>These bits are effective in SCI compatible mode only. These bits set the SCI character length from 1 to 8 bits.</p> <p>Note: In compatibility mode or buffered SCI mode, when data of fewer than eight bits in length is received, it is left justified in SCIRD/RDy and padded with trailing zeros. Data read from the SCIRD should be shifted by software to make the received data right justified.</p> <p>Note: Data written to the SCITD should be right justified but does not need to be padded with leading zeros.</p> <p>These bits are writable in SCI mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The character is 1 bits long.            1h (R/W) = The character is 2 bits long.            2h (R/W) = The character is 3 bits long.            3h (R/W) = The character is 4 bits long.            4h (R/W) = The character is 5 bits long.            5h (R/W) = The character is 6 bits long.            6h (R/W) = The character is 7 bits long.            7h (R/W) = The character is 8 bits long.</p>

**27.6.2.12 BRSR Register (Offset = 2Ch) [reset = 0h]**

BRSR is shown in [Figure 27-38](#) and described in [Table 27-27](#).

Return to the [Summary Table](#).

The BRSR register is used to configure the baud rate of the LIN module.

**Figure 27-38. BRSR Register**

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

**Table 27-27. BRSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30-28	U	R/W	0h	Superfractional Divider Selection. (U) These bits are an additional fractional part for the baudrate specification. These bits allow a super fine tuning of the fractional baudrate with 7 more intermediate values for each of the M fractional divider values. See the Superfractional Divider section for more details. Reset type: SYSRSn
27-24	M	R/W	0h	SCI/LIN 4-bit Fractional Divider Selection. (M) These bits are effective in LIN or SCI asynchronous mode. These bits are used to select a baud rate for the SCI/LIN module, and they are a fractional part for the baud rate specification. The M divider allows fine-tuning of the baud rate over the P prescaler with 15 additional intermediate values for each of the P integer values. Reset type: SYSRSn
23-16	SCI_LIN_PSH	R/W	0h	PRESCALER P (High Bits). SCI/LIN 24-bit Integer Prescaler Selection. These bits are used to select a baudrate for the SCI/LIN module. These bits are effective in LIN mode and SCI compatible mode. The SCI/LIN has an internally generated serial clock determined by the LIN module input clock and the prescalers P and M in this register. The SCI/LIN uses the 24-bit integer prescaler P value to select 1 of over 16,700,000 available baud rates. The additional 4-bit fractional prescaler M refines the baudate selection. Reset type: SYSRSn

**Table 27-27. BRSR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15-0	SCI_LIN_PSL	R/W	0h	<p>PRESCALER P (Low Bits). SCI/LIN 24-bit Integer Prescaler Selection.</p> <p>These bits are used to select a baudrate for the SCI/LIN module. These bits are effective in LIN mode and SCI compatible mode. The SCI/LIN has an internally generated serial clock determined by the LIN module input clock and the prescalers P and M in this register. The SCI/LIN uses the 24-bit integer prescaler P value to select 1 of over 16,700,000 available baud rates. The additional 4-bit fractional prescaler M refines the baudrate selection.</p> <p>Reset type: SYSRSn</p>

### 27.6.2.13 SCIED Register (Offset = 30h) [reset = 0h]

SCIED is shown in [Figure 27-39](#) and described in [Table 27-28](#).

Return to the [Summary Table](#).

The SCIED register is a duplicate copy of SCIRD register that has no affect on the RXRDY flag for use with an emulator.

**Figure 27-39. SCIED 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																ED															
R-0h																R-0h															

**Table 27-28. SCIED Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ED	R	0h	Receiver Emulation Data. This bit is effective in SCI-compatible mode only. Reading SCIED(7-0) does not clear the RXRDY flag. This register should be used only by an emulator that must continually read the data buffer without affecting the RXRDY flag. Reset type: SYSRSn



**27.6.2.14 SCIRD Register (Offset = 34h) [reset = 0h]**

SCIRD is shown in [Figure 27-40](#) and described in [Table 27-29](#).

Return to the [Summary Table](#).

The SCIRD register is where received data is stored and can be read from.

**Figure 27-40. SCIRD 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														RD																	
R-0h														R-0h																	

**Table 27-29. SCIRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	RD	R	0h	<p>Received Data.</p> <p>This bit is effective in SCI-compatible mode only. When a frame has been completely received, the data in the frame is transferred from the receiver shift register SCIRXSHF to this register. As this transfer occurs, the RXRDY flag is set and a receive interrupt is generated if RX INT ENA (SCISSETINT0.9) is set. When the data is read from SCIRD, the RXRDY flag is automatically cleared.</p> <p>When the SCI receives data that is fewer than eight bits in length, it loads the data into this register in a left justified format padded with trailing zeros. Therefore, your software should perform a logical shift on the data by the correct number of positions to make it right justified.</p> <p>Reset type: SYSRSn</p>

**27.6.2.15 SCITD Register (Offset = 38h) [reset = 0h]**

SCITD is shown in [Figure 27-41](#) and described in [Table 27-30](#).

Return to the [Summary Table](#).

The SCITD register is where data to be transmitted is written to by application software.

**Figure 27-41. SCITD 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																								TD							
R-0h																								R/W-0h							

**Table 27-30. SCITD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	TD	R/W	0h	Transmit data This bit is effective in SCI-compatible mode only. Data to be transmitted is written to this register. The transfer of data from this register to the transmit shift register SCITXSHF sets the TXRDY flag (SCIFLR.23), which indicates that SCITD is ready to be loaded with another byte of data. Note: If TX INT ENA (SCISSETINT.8) is set, this data transfer also causes an interrupt. Note: Data written to the SCIRD register that is fewer than eight bits long must be right justified, but it does not need to be padded with leading zeros. Reset type: SYSRSn

**27.6.2.16 SCIPIO0 Register (Offset = 3Ch) [reset = 0h]**

SCIPIO0 is shown in [Figure 27-42](#) and described in [Table 27-31](#).

Return to the [Summary Table](#).

The SCIPIO0 register is used to enable the LINTX and LINRX pins.

**Figure 27-42. SCIPIO0 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					TXFUNC	RXFUNC	RESERVED
R-0h					R/W-0h	R/W-0h	R-0h

**Table 27-31. SCIPIO0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-3	RESERVED	R	0h	Reserved
2	TXFUNC	R/W	0h	Transmit pin function. This bit is effective in LIN or SCI mode. This bit defines the function of LINTX pin. Reset type: SYSRSn 0h (R/W) = LINTX pin is disabled. 1h (R/W) = LINTX pin is enabled.
1	RXFUNC	R/W	0h	Receive pin function. This bit is effective in LIN or SCI mode. This bit defines the function of the LINRX pin. Reset type: SYSRSn 0h (R/W) = LINRX pin is disabled. 1h (R/W) = LINRX pin is enabled.
0	RESERVED	R	0h	Reserved

**27.6.2.17 SCIPIO2 Register (Offset = 44h) [reset = 0h]**

SCIPIO2 is shown in [Figure 27-43](#) and described in [Table 27-32](#).

Return to the [Summary Table](#).

The SCIPIO2 register indicates the current status of the LINTX and LINRX pins.

**Figure 27-43. SCIPIO2 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					TXIN	RXIN	RESERVED
R-0h					R-0h	R-0h	R-0h

**Table 27-32. SCIPIO2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-3	RESERVED	R	0h	Reserved
2	TXIN	R	0h	Transmit data in. This bit is effective in LIN or SCI-compatible mode. This bit contains the current value on the LINTX pin. Reset type: SYSRSn
1	RXIN	R	0h	Receive data in. This bit is effective in LIN or SCI-compatible mode. This bit contains the current value on the LINRX pin. Reset type: SYSRSn
0	RESERVED	R	0h	Reserved

**27.6.2.18 LINCOMP Register (Offset = 60h) [reset = 0h]**

LINCOMP is shown in [Figure 27-44](#) and described in [Table 27-33](#).

Return to the [Summary Table](#).

The LINCOMPARE register is used to configure the sync delimiter and sync break extension.

**Figure 27-44. LINCOMP 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						SDEL		RESERVED						SBREAK	
R-0h						R/W-0h		R-0h						R/W-0h	

**Table 27-33. LINCOMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-10	RESERVED	R	0h	Reserved
9-8	SDEL	R/W	0h	<p>2-bit Sync Delimiter compare.</p> <p>These bits are effective in LIN mode only. These bits are used to configure the number of Tbit for the sync delimiter in the sync field. The time delay calculation for the synchronization delimiter is:  <math>TSDEL = (SDEL + 1)Tbit</math></p> <p>These bits are writable in LIN mode only.  Reset type: SYSRSn  0h (R/W) = The sync delimiter has 1 Tbit.  1h (R/W) = The sync delimiter has 2 Tbit.  2h (R/W) = The sync delimiter has 3 Tbit.  3h (R/W) = The sync delimiter has 4 Tbit.</p>
7-3	RESERVED	R	0h	Reserved
2-0	SBREAK	R/W	0h	<p>3-bit Sync Break extend.</p> <p>LIN mode only. These bits are used to configure the number of Tbits for the sync break to extend the minimum 13 Tbit in the Sync Field to a maximum of 20 Tbit. The time delay calculation for the sync break is:  <math>TSYNBRK = 13Tbit + SBREAK \times Tbit</math></p> <p>These bits are writable in LIN mode only.  Reset type: SYSRSn  0h (R/W) = The sync break has no additional Tbit.  1h (R/W) = The sync break has 1 additional Tbit.  2h (R/W) = The sync break has 2 additional Tbit.  3h (R/W) = The sync break has 3 additional Tbit.  4h (R/W) = The sync break has 4 additional Tbit.  5h (R/W) = The sync break has 5 additional Tbit.  6h (R/W) = The sync break has 6 additional Tbit.  7h (R/W) = The sync break has 7 additional Tbit.</p>

**27.6.2.19 LINRD0 Register (Offset = 64h) [reset = 0h]**

LINRD0 is shown in [Figure 27-45](#) and described in [Table 27-34](#).

Return to the [Summary Table](#).

The LINRD0 register contains the lower 4 bytes of the received LIN frame data.

**Figure 27-45. LINRD0 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
RD0								RD1								RD2								RD3							
R-0h								R-0h								R-0h								R-0h							

**Table 27-34. LINRD0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RD0	R	0h	8-bit Receive Buffer 0 Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received. A read of this byte clears the RXDY byte. Note: RD<x-1> is equivalent to Data byte <x> of the LIN frame. Reset type: SYSRSn
23-16	RD1	R	0h	8-bit Receive Buffer 1. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received. Reset type: SYSRSn
15-8	RD2	R	0h	8-bit Receive Buffer 2. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received. Reset type: SYSRSn
7-0	RD3	R	0h	8-bit Receive Buffer 3. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received. Reset type: SYSRSn

**27.6.2.20 LINRD1 Register (Offset = 68h) [reset = 0h]**

LINRD1 is shown in [Figure 27-46](#) and described in [Table 27-35](#).

Return to the [Summary Table](#).

The LINRD1 register contains the upper 4 bytes of the received LIN frame data.

**Figure 27-46. LINRD1 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
RD4								RD5								RD6								RD7							
R-0h								R-0h								R-0h								R-0h							

**Table 27-35. LINRD1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RD4	R	0h	8-bit Receive Buffer 4 Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received. Reset type: SYSRSn
23-16	RD5	R	0h	8-bit Receive Buffer 5. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received. Reset type: SYSRSn
15-8	RD6	R	0h	8-bit Receive Buffer 6. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received. Reset type: SYSRSn
7-0	RD7	R	0h	8-bit Receive Buffer 7. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received. Reset type: SYSRSn

**27.6.2.21 LINMASK Register (Offset = 6Ch) [reset = 0h]**

LINMASK is shown in [Figure 27-47](#) and described in [Table 27-36](#).

Return to the [Summary Table](#).

The LINMASK register is used to configure the masks used for filtering incoming ID messages for receive and transmit frames.

**Figure 27-47. LINMASK 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								RXIDMASK								RESERVED								TXIDMASK							
R-0h								R/W-0h								R-0h								R/W-0h							

**Table 27-36. LINMASK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	RXIDMASK	R/W	0h	Receive ID mask. This field is effective in LIN mode only. This 8-bit mask is used for filtering an incoming ID message and compare it to the ID-byte. A compare match of the received ID with the RX ID mask will set the ID RX flag and trigger an ID interrupt if enabled. A 0 bit in the mask indicates that that bit is compared to the ID-byte. A 1 bit in the mask indicates that that bit is filtered and therefore not used in the compare. When HGENCTRL is set to 1, this field must be set to 0xFF. Reset type: SYSRSn
15-8	RESERVED	R	0h	Reserved
7-0	TXIDMASK	R/W	0h	Transmit ID mask. This field is effective in LIN mode only. This 8-bit mask is used for filtering an incoming ID message and compare it to the ID-byte. A compare match of the received ID with the TX ID Mask will set the ID TX flag and trigger an ID interrupt if enabled. A 0 bit in the mask indicates that bit is compared to the ID-byte. A 1 bit in the mask indicates that bit is filtered and therefore not used for the compare. When HGENCTRL is set to 1, this field must be set to 0xFF. Reset type: SYSRSn



**27.6.2.22 LINID Register (Offset = 70h) [reset = 0h]**

LINID is shown in [Figure 27-48](#) and described in [Table 27-37](#).

Return to the [Summary Table](#).

The LINID register contains the identification fields for LIN communication.

NOTE: For software compatibility with future LIN modules, the HGEN CTRL bit must be set to 1, the RX ID MASK field must be set to FFh, and the TX ID MASK field must be set to FFh.

**Figure 27-48. LINID Register**

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

**Table 27-37. LINID Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	RECEIVEDID	R	0h	Received ID. This bit is effective in LIN mode only. This byte contains the current message identifier. During header reception the received ID is copied from the SCIRXSHF register to this byte if there is no ID-parity error and there has been an RX/TX match. Note: If a framing error (FE) is detected during ID reception, the received ID will also not be copied to the LINID register. Reset type: SYSRSn
15-8	IDSLAVETASKBYTE	R/W	0h	ID Slave Task byte. This field is effective in LIN mode only. This byte contains the identifier to which the received ID of an incoming header will be compared in order to decide whether a RX response, a TX response, or no action needs to be done by the LIN node. These bits are writable in LIN mode only. Reset type: SYSRSn
7-0	IDBYTE	R/W	0h	ID byte. This field is effective in LIN mode only. This byte is the LIN mode message ID. On a master node, a write to this register by the CPU initiates a header transmission. For a slave task, this byte is used for message filtering when HGENCTRL (SCIGCR1.12) is '0'. These bits are writable in LIN mode only. Reset type: SYSRSn

### 27.6.2.23 LINTD0 Register (Offset = 74h) [reset = 0h]

LINTD0 is shown in [Figure 27-49](#) and described in [Table 27-38](#).

Return to the [Summary Table](#).

The LINTD0 register contains the lower 4 bytes of the data to be transmitted.

NOTE: TD<x-1> is equivalent to Data byte <x> of the LIN frame.

**Figure 27-49. LINTD0 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
TD0								TD1								TD2								TD3							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 27-38. LINTD0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	TD0	R/W	0h	8-bit Transmit Buffer 0. Byte 0 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TDO buffer, transmission will be initiated. Reset type: SYSRSn
23-16	TD1	R/W	0h	8-bit Transmit Buffer 3. Byte 1 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Reset type: SYSRSn
15-8	TD2	R/W	0h	8-bit Transmit Buffer 2. Byte 2 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Reset type: SYSRSn
7-0	TD3	R/W	0h	8-bit Transmit Buffer 3. Byte 3 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Reset type: SYSRSn

### 27.6.2.24 LINTD1 Register (Offset = 78h) [reset = 0h]

LINTD1 is shown in [Figure 27-50](#) and described in [Table 27-39](#).

Return to the [Summary Table](#).

The LINTD1 register contains the upper 4 bytes of the data to be transmitted.

NOTE: TD<x-1> is equivalent to Data byte <x> of the LIN frame.

**Figure 27-50. LINTD1 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
TD4								TD5								TD6								TD7							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 27-39. LINTD1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	TD4	R/W	0h	8-bit Transmit Buffer 4. Byte 4 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Reset type: SYSRSn
23-16	TD5	R/W	0h	8-bit Transmit Buffer 5. Byte 5 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Reset type: SYSRSn
15-8	TD6	R/W	0h	8-bit Transmit Buffer 6. Byte 6 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Reset type: SYSRSn
7-0	TD7	R/W	0h	8-bit Transmit Buffer 7. Byte 7 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Reset type: SYSRSn

**27.6.2.25 MBRSR Register (Offset = 7Ch) [reset = DACH]**

MBRSR is shown in [Figure 27-51](#) and described in [Table 27-40](#).

Return to the [Summary Table](#).

The MBRSR register is used to configure the expected maximum baud rate of the LIN network.

**Figure 27-51. MBRSR 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													MBR																		
R-0h													R/W-DACH																		

**Table 27-40. MBRSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	Reserved
12-0	MBR	R/W	DACH	<p>Maximum Baud Rate Prescaler.</p> <p>This field is effective in LIN mode only. This 13-bit prescaler is used during the synchronization phase (see the "Header Reception and Adaptive Baudrate" section) of a slave module if the ADAPT bit is set. In this way, a SCI/LIN slave using an automatic or select bit rate modes detects any LIN bus legal rate automatically.</p> <p>The MBR value should be programmed to allow a maximum baud rate that is not more than 10% above the expected operating baud rate in the LIN network. Otherwise a s 0x00 data byte could mistakenly be detected as sync break.</p> <p>The default value is for a 70MHz LINCLK (0xDAC).</p> <p>This MBR prescaler is used by the wake-up and idle time counters for a constant expiration time relative to a 20kHz rate.</p> <p>Reset type: SYSRSn</p>

### 27.6.2.26 IODFTCTRL Register (Offset = 90h) [reset = 500h]

IODFTCTRL is shown in [Figure 27-52](#) and described in [Table 27-41](#).

Return to the [Summary Table](#).

The IODFTCTRL register is used to emulate various error and test conditions.

**Figure 27-52. IODFTCTRL Register**

31	30	29	28	27	26	25	24
BERRENA	PBERRENA	CERRENA	ISFERRENA	RESERVED	FERRENA	PERRENA	BRKDTERRENA
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED			PINSAMPLEMASK		TXSHIFT		
R/W-0h			R/W-0h		R/W-0h		
15	14	13	12	11	10	9	8
RESERVED				IODFTENA			
R-0h				R/W-5h			
7	6	5	4	3	2	1	0
RESERVED						LPBENA	RXPENA
R-0h						R/W-0h	R/W-0h

**Table 27-41. IODFTCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	BERRENA	R/W	0h	Bit Error Enable bit. This bit is effective in LIN mode only. This bit is used to create a Bit error. When this bit is set, the bit received is ORed with 1 and passed to the Bit monitor circuitry. Reset type: SYSRSn
30	PBERRENA	R/W	0h	Physical Bus Error Enable bit. This bit is effective in LIN mode only. This bit is used to create a Physical Bus Error. When this bit is set, the bit received during Sync Break field transmission is ORed with 1 and passed to the Bit monitor circuitry. Reset type: SYSRSn
29	CERRENA	R/W	0h	Checksum Error Enable bit. This bit is effective in LIN mode only. This bit is used to create a checksum error. When this bit is set, the polarity of the CTYPE (checksum type) in the receive checksum calculator is changed so that a checksum error is generated. Reset type: SYSRSn
28	ISFERRENA	R/W	0h	Inconsistent Sync Field Error Enable bit. This bit is effective in LIN mode only. This bit is used to create an ISF error. When this bit is set, the bit widths in the sync field are varied so that the ISF check fails and the error flag is set. Reset type: SYSRSn
27	RESERVED	R	0h	Reserved
26	FERRENA	R/W	0h	This bit is used to create a Frame Error. This bit is effective in SCI-compatible mode only. When this bit is set, the stop bit received is ANDed with '0' and passed to the stop bit check circuitry. Reset type: SYSRSn

**Table 27-41. IODFTCTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25	PERRENA	R/W	0h	Compatible Mode only This bit is effective in SCI-compatible mode only. This bit is used to create a Parity Error. When this bit is set, in compatible mode, the parity bit received is toggled so that a parity error occurs. Reset type: SYSRSn
24	BRKDTERRNA	R/W	0h	Compatible Mode only This bit is effective in SCI-compatible mode only. This bit is used to create BRKDT error (SCI mode only). When this bit is set, the stop bit of the frame is ANDed with '0' and passed to the RSM so that a frame error occurs. Then the RX Pin is forced to continuous low for 10 Tbits so that a BRKDT error occurs. Reset type: SYSRSn
23-21	RESERVED	R/W	0h	Reserved
20-19	PINSAMPLEMASK	R/W	0h	Pin sample mask. These bits define the sample number at which the TX Pin value that is being transmitted will be inverted to verify the receive pin samples correctly with the majority detection circuitry. Note: During IODFT mode testing for the pin sample mask, the prescaler P must be programmed to be greater than 2. Reset type: SYSRSn 0h (R/W) = No Mask 1h (R/W) = Invert the TX Pin value at TBIT_CENTER 2h (R/W) = Invert the TX Pin value at TBIT_CENTER + SCLK 3h (R/W) = Invert the TX Pin value at TBIT_CENTER + 2 SCLK
18-16	TXSHIFT	R/W	0h	Transmit shift. These bits define the delay by which the value on LINTX is delayed so that the value on LINRX is asynchronous. (Not applicable to Start Bit) Reset type: SYSRSn 0h (R/W) = No Delay 1h (R/W) = Delay by 1 SCLK 2h (R/W) = Delay by 2 SCLK 3h (R/W) = Delay by 3 SCLK 4h (R/W) = Delay by 4 SCLK 5h (R/W) = Delay by 5 SCLK 6h (R/W) = Delay by 6 SCLK 7h (R/W) = Delay by 7 SCLK
15-12	RESERVED	R	0h	Reserved

**Table 27-41. IODFTCTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-8	IODFTENA	R/W	5h	<p>IO DFT Enable Key</p> <p>This field is used to enable the IODFT mode of the SCI/LIN module for testing.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = IODFT is disabled</p> <p>1h (R/W) = IODFT is disabled</p> <p>2h (R/W) = IODFT is disabled</p> <p>3h (R/W) = IODFT is disabled</p> <p>4h (R/W) = IODFT is disabled</p> <p>5h (R/W) = IODFT is disabled</p> <p>6h (R/W) = IODFT is disabled</p> <p>7h (R/W) = IODFT is disabled</p> <p>8h (R/W) = IODFT is disabled</p> <p>9h (R/W) = IODFT is disabled</p> <p>Ah (R/W) = IODFT is enabled</p> <p>Bh (R/W) = IODFT is disabled</p> <p>Ch (R/W) = IODFT is disabled</p> <p>Dh (R/W) = IODFT is disabled</p> <p>Eh (R/W) = IODFT is disabled</p> <p>Fh (R/W) = IODFT is disabled</p>
7-2	RESERVED	R	0h	Reserved
1	LPBENA	R/W	0h	<p>Module loopback enable.</p> <p>In analog loopback mode the complete communication path through the I/Os can be tested, whereas in digital loopback mode the I/O buffers are excluded from this path.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Digital loopback is enabled.</p> <p>1h (R/W) = Analog loopback is enabled in module I/O DFT mode (when IODFTENA = 1010)</p>
0	RXPENA	R/W	0h	<p>Module Analog loopback through receive pin enable.</p> <p>This bit defines whether the I/O buffers for the transmit or the receive pin are included in the communication path in analog loopback mode only.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Analog loopback through the transmit pin is enabled.</p> <p>1h (R/W) = Analog loopback through the receive pin is enabled.</p>

**27.6.2.27 LIN\_GLB\_INT\_EN Register (Offset = E0h) [reset = 0h]**

LIN\_GLB\_INT\_EN is shown in [Figure 27-53](#) and described in [Table 27-42](#).

Return to the [Summary Table](#).

The LIN\_GLB\_INT\_EN register is used to enable the INT0 and INT1 interrupt lines to propagate to the PIE block.

**Figure 27-53. LIN\_GLB\_INT\_EN 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						GLBINT1_EN	GLBINT0_EN
R-0h						R/W-0h	R/W-0h

**Table 27-42. LIN\_GLB\_INT\_EN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	GLBINT1_EN	R/W	0h	Global Interrupt Enable for LIN INT1. This bit determines whether the INT1 interrupt line generates an interrupt to the PIE or not. Reset type: SYSRSn 0h (R/W) = LIN INT1 line does not generate an interrupt to the PIE. 1h (R/W) = LIN INT1 line generates an interrupt to the PIE if an enabled interrupt condition occurs.
0	GLBINT0_EN	R/W	0h	Global Interrupt Enable for LIN INT0. This bit determines whether the INT0 interrupt line generates an interrupt to the PIE or not. Reset type: SYSRSn 0h (R/W) = LIN INT0 line does not generate an interrupt to the PIE. 1h (R/W) = LIN INT0 line generates an interrupt to the PIE if an enabled interrupt condition occurs.



**27.6.2.28 LIN\_GLB\_INT\_FLG Register (Offset = E4h) [reset = 0h]**

LIN\_GLB\_INT\_FLG is shown in [Figure 27-54](#) and described in [Table 27-43](#).

Return to the [Summary Table](#).

The LIN\_GLB\_INT\_FLG register contains the current status of the INT0 and INT1 flags.

**Figure 27-54. LIN\_GLB\_INT\_FLG 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						INT1_FLG	INT0_FLG
R-0h						R-0h	R-0h

**Table 27-43. LIN\_GLB\_INT\_FLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	INT1_FLG	R	0h	Global Interrupt Flag for LIN INT1. This bit indicates if an interrupt was generated to the PIE due to an enabled interrupt on the INT1 interrupt line. Refer to the LIN Interrupt Status Register for the condition that generated the interrupt. This bit can be cleared by writing a 1 to the corresponding bit in the LIN_GLB_INT_CLR register. Reset type: SYSRSn 0h (R/W) = No interrupt is active on the INT1 line. 1h (R/W) = An interrupt was generated due to an enabled interrupt on the INT1 interrupt line.
0	INT0_FLG	R	0h	Global Interrupt Flag for LIN INT0. This bit indicates if an interrupt was generated to the PIE due to an enabled interrupt on the INT0 interrupt line. Refer to the LIN Interrupt Status Register for the condition that generated the interrupt. This bit can be cleared by writing a 1 to the corresponding bit in the LIN_GLB_INT_CLR register. Reset type: SYSRSn 0h (R/W) = No interrupt is active on the INT0 line. 1h (R/W) = An interrupt was generated due to an enabled interrupt on the INT0 interrupt line.

**27.6.2.29 LIN\_GLB\_INT\_CLR Register (Offset = E8h) [reset = 0h]**

LIN\_GLB\_INT\_CLR is shown in [Figure 27-55](#) and described in [Table 27-44](#).

Return to the [Summary Table](#).

The LIN\_GLB\_INT\_CLR register is used to clear the interrupt flags in LIN\_GLB\_INT\_FLG register.

**Figure 27-55. LIN\_GLB\_INT\_CLR 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						INT1_FLG_CLR	INT0_FLG_CLR
R-0h						R	R
R-0h						R/W1C-0h	R/W1C-0h

**Table 27-44. LIN\_GLB\_INT\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	INT1_FLG_CLR	R/W1C	0h	Global Interrupt flag clear for LIN INT1. This bit is used to clear the corresponding bit in the LIN_GLB_INT_FLG register. Write 1 to clear the INT1_FLG bit. Writing 0 has no effect. Reset type: SYSRSn
0	INT0_FLG_CLR	R/W1C	0h	Global Interrupt flag clear for LIN INT0. This bit is used to clear the corresponding bit in the LIN_GLB_INT_FLG register. Write 1 to clear the INT0_FLG bit. Writing 0 has no effect. Reset type: SYSRSn

## Fast Serial Interface (FSI)

---

---

This chapter contains a general description of the Fast Serial Interface (FSI) module. The FSI is a serial peripheral capable of reliable high-speed communication across isolation barriers.

Topic	Page
<b>28.1 Introduction .....</b>	<b>2580</b>
<b>28.2 Features .....</b>	<b>2580</b>
<b>28.3 System-level Integration .....</b>	<b>2580</b>
<b>28.4 FSI Operation .....</b>	<b>2586</b>
<b>28.5 Programmer's Model .....</b>	<b>2607</b>
<b>28.6 FSI Registers .....</b>	<b>2610</b>
<b>28.7 Register to Driverlib Function Mapping .....</b>	<b>2680</b>

## 28.1 Introduction

The Fast Serial Interface (FSI) module is a serial communication peripheral capable of reliable high-speed communication across isolation devices. Galvanic isolation devices are used in situations where two different electronic circuits, which do not have common power and ground connections, must exchange information. Though isolation devices facilitate these signal communications, they can also introduce a large delay on the signal lines and add skew between the signals. The FSI is designed specifically to ensure reliable high-speed communication for system scenarios that involve communication across isolation barriers without adding components.

The FSI consists of independent transmitter (FSITX) and receiver (FSIRX) cores. The FSITX and FSIRX cores are configured and operated independently.

For additional information on the FSI module, please refer to the following source:

- <http://www.ti.com/lit/an/spracj9/spracj9.pdf>

## 28.2 Features

The FSI module includes the following features:

- Independent transmitter and receiver cores
- Source-synchronous transmission
- Double Data Rate (DDR)
- One or two data lines
- Programmable data length
- Skew adjustment block to compensate for board and system delay mismatches
- Frame error detection
- Programmable frame tagging for message filtering
- Hardware ping to detect line breaks during communication (ping watchdog)
- Two interrupts per FSI core
- Externally triggered frame generation
- Hardware- or software-calculated CRC
- Embedded ECC computation module
- Register write protection
- DMA support
- CLA task triggering
- SPI compatibility mode (limited features available)

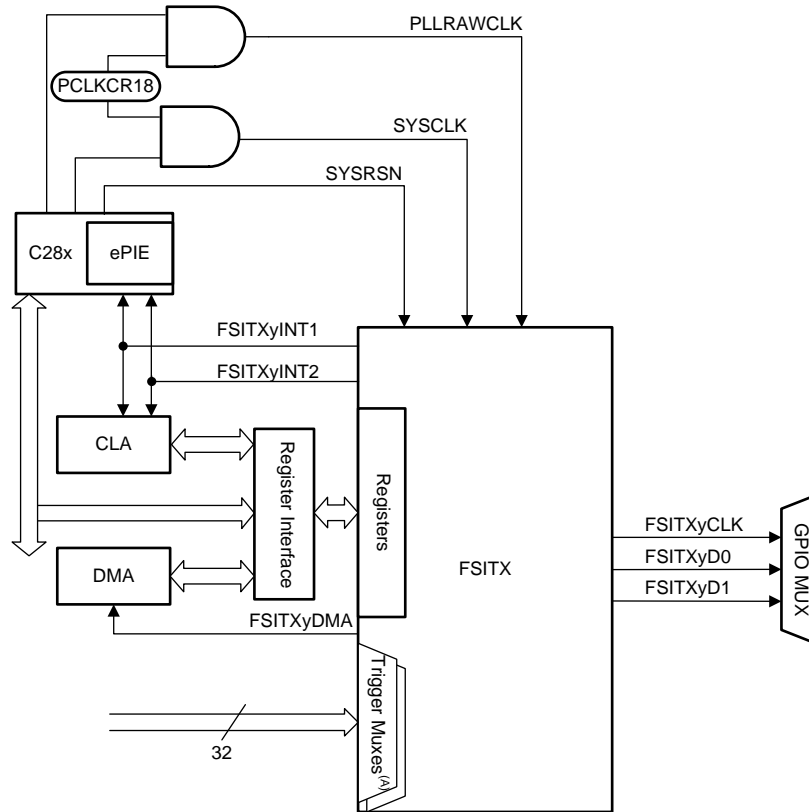
## 28.3 System-level Integration

This section describes the device-level integration of the FSI module. Some of the features may require additional configuration of modules that are not within the scope of this chapter, the details of which can be found elsewhere in this document.

### 28.3.1 CPU Interface

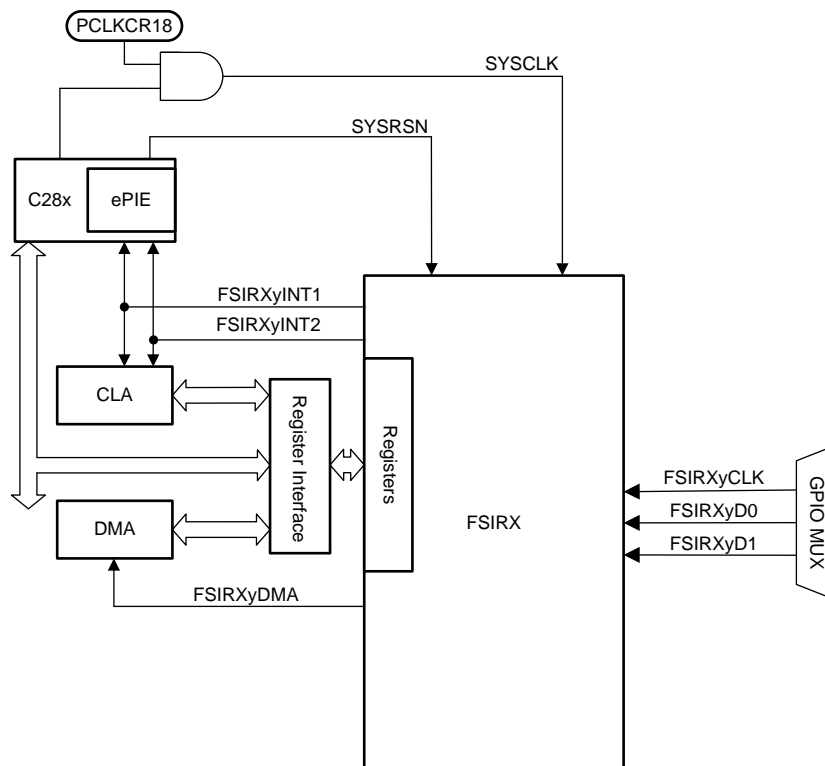
The following diagrams show the CPU interface of each FSI module.

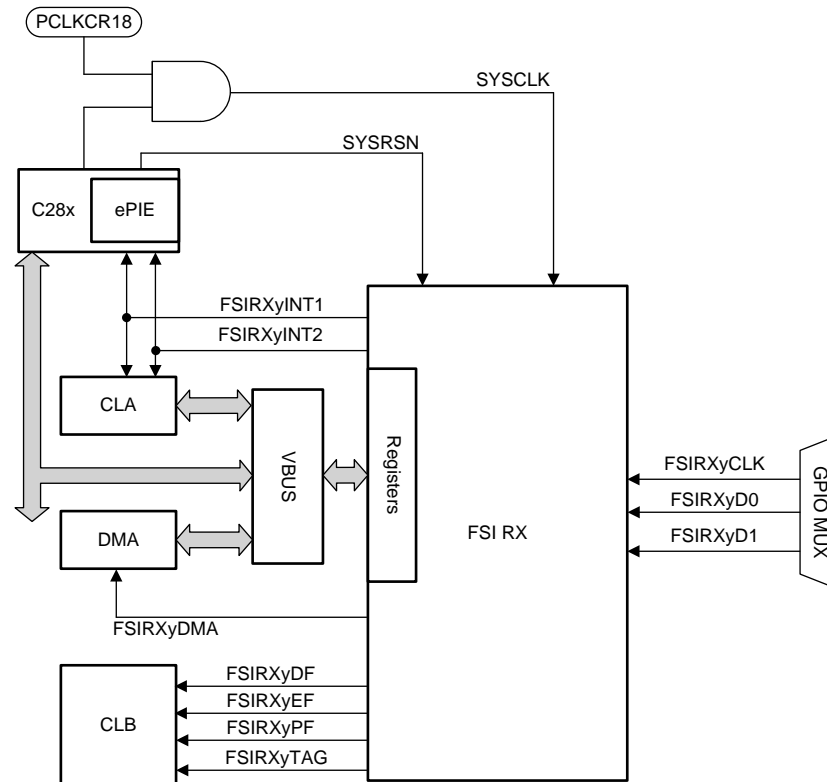
**Figure 28-1. FSI Transmitter (FSITX) CPU Interface**



A The signals connected to the trigger muxes are described in [Section 28.3.6](#).

**Figure 28-2. FSI Receiver (FSIRX) CPU Interface**



**Figure 28-3. FSI Receiver (FSIRX) CPU Interface with CLB**


### 28.3.2 Signal Description

FSI is a point-to-point communication protocol. Hence, an FSI transmitter core will communicate directly to a single FSI receiver core. Similarly, an FSI receiver core will receive data from a single FSI transmitter core.

Each FSI core has three signals associated with it: one clock and two data signals. Data is always transmitted or received with the most significant bit of each frame field being first. If multi-lane transmissions are not used, the TXD1 and RXD1 signals can be left unconnected and their GPIOs repurposed for other application needs. [Table 28-1](#) and [Table 28-2](#) describe the various signals that can be selected by the GPIO mux to be brought out to device pins.

**Table 28-1. FSI Transmitter Core Signals**

Signal Name	Direction	Description	Inactive Level <sup>(1)</sup>
TXCLK	Output	<p>This is the transmit clock. It is driven by the FSI transmit module.</p> <p>During a transmission, four clock edges will be transmitted before the start of frame phase (preamble) and four clock edges will follow the last bit of the frame (postamble). Data is transmitted on both edges of the clock.</p> <p>In SPI compatibility mode, the preamble and the post frame clock edges will not be transmitted. Data is transmitted only on one edge of the clock. The data is launched on the falling edge of the clock with the expectation that the receiving SPI will capture the data on the rising edge.</p>	Logic High
TXD0	Output	This is the primary data output line for transmission. This signal is driven by the FSI transmit module.	Logic High

<sup>(1)</sup> Inactive level refers to the state of the pin while the module is not actively transmitting, or held in reset.

**Table 28-1. FSI Transmitter Core Signals (continued)**

Signal Name	Direction	Description	Inactive Level <sup>(1)</sup>
		When the FSI is configured for multi-lane transmission, TXD0 will contain all the even numbered bits of the data and CRC bytes. Other frame fields such as frame type, start-of-frame, tag, and end-of-frame will be transmitted in full.	
TXD1	Output	This is an additional data output line for transmission if the FSI is configured for multi-lane transmission. This signal is driven by the FSI transmit module.  During transmission, the data bits are split between TXD0 and TXD1. TXD1 will contain all the odd numbered bits of the data and CRC bytes. This applies only to the data words and the CRC bytes. Other data frame related information like Frame Type, Start-of-Frame, Tag and End-of-frame, the state of this line will be identical to TXD0.	Logic High

**Table 28-2. FSI Receiver Core Signals**

Signal Name	Direction	Description	Inactive Level <sup>(1)</sup>
RXCLK	Input	This is the receive clock input signal for the FSI receive module.  This must should be connected to TXCLK of the transmitting FSI module.	Logic High
RXD0	Input	This is the primary data input line for reception. This should be connected to the TXD0 of the transmitting FSI module.	Logic High
RXD1	Input	This is an additional data input line for reception. This signal should be connected to the TXD1 of the transmitting FSI module if multi-lane transmission is used.	Logic High

<sup>(1)</sup> Inactive level refers to the state of the pin while the module is not actively receiving data.

### 28.3.2.1 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 0x3. The internal pullups can be configured in the GPyPUD register. See *General Purpose Input-Output*, [Chapter 8](#), for more details on the GPIO mux and settings.

### 28.3.3 Interrupts

Each FSI module contains multiple interrupt sources which can be assigned to two different interrupt vectors: INT1 and INT2. Each interrupt source has an associated status flag, force, and clear bits in the EVT\_STS, EVT\_FRC, and the EVT\_CLR registers, respectively.

---

**NOTE:** Because the transmitter and receiver cores have their own distinct register sets, the preceding 'TX\_' and 'RX\_' will be left off of the register names unless otherwise noted.

---

Each interrupt can be assigned to either interrupt vector, INT1 and INT2, to allow for two priority levels. Alternately, the interrupt source can be prevented from generating any interrupt, though the status flag can still be set and monitored by software. The transmitter events are assigned to either interrupt vector in the TX\_INT\_CTRL register. The receiver events are assigned an interrupt vector using RX\_INT1\_CTRL and RX\_INT2\_CTRL registers. If an interrupt is not required, ensure the bit is not set in the respective INT\_CTRL register.

#### 28.3.3.1 Transmitter Interrupts

The transmitter can generate the following interrupts:

- **Frame Done (FRAME\_DONE)**  
This event indicates that FSI has completed transmitting a frame.
- **Buffer Underrun (BUF\_UNDERRUN)**  
This event indicates that the transmit buffer has experienced underrun. Buffer underrun occurs when the transmitter tries to read data from a location which has not yet be written to by the CLA, CPU, or DMA.
- **Buffer Overrun (BUF\_OVERRUN)**  
The buffer overrun interrupt is generated when the buffer has experienced overrun. Buffer overrun may occur if a piece of data is overwritten before it has been transmitted.
- **Ping Frame Triggered (PING\_TRIGGERED)**  
The ping frame triggered interrupt is generated when the ping frame has been triggered. This bit will be set when the ping counter has timed out or an external ping trigger event has occurred.

### 28.3.3.2 Receiver Interrupts

The receiver core is capable of generating interrupts from many different events. These events are described below.

- **Ping Watchdog Timeout (PING\_WD\_TO)**  
This event indicates that the ping watchdog timer has timed out. The receiver has not received a valid frame within the time period specified in the RX\_PING\_WD\_REF register.
- **Frame Watchdog Timeout (FRAME\_WD\_TO)**  
This event indicates that the frame watchdog timer has timed out. The conditions of this timeout are set using the RX\_FRAME\_WD\_CTRL register. As soon as the start of frame phase is detected, the frame watchdog counter will start counting from 0. The end of frame phase must complete by the time the watchdog counter reaches the reference value. If this does not happen, the watchdog will time out and this event will be generated. If this event occurs, the receiver must undergo a soft reset and subsequent resynchronization in order to guarantee proper operation.
- **CRC Error (CRC\_ERR)**  
This error indicates that a CRC error has occurred. A CRC error will be generated when the received CRC and the computed CRC do not match.
- **Frame Type Error (TYPE\_ERR)**  
This error indicates that an invalid frame type has been received. If this error occurs, the receiver must undergo a soft reset and subsequent resynchronization in order to guarantee proper operation.
- **End-of-Frame Error (EOF\_ERR)**  
This error indicates that an invalid end-of-frame bit pattern has been received. If this error occurs, the receiver must undergo a soft reset and subsequent resynchronization in order to guarantee proper operation.
- **Receive Buffer Overrun (BUF\_OVERRUN)**  
This event indicates that an overrun condition has occurred in the receive buffer.
- **Receive Buffer Underrun (BUF\_UNDERRUN)**  
This event indicates that an underrun condition has occurred in the receive buffer. This condition occurs when software reads the buffer while it is empty.
- **Frame Done (FRAME\_DONE)**  
This event indicates that a valid frame has been received without error.
- **Error Frame Received (ERR\_FRAME)**  
This event indicates that an error frame has been received.
- **Ping Frame Received (PING\_FRAME)**  
This event indicates that a ping frame has been received.
- **Frame Overrun (FRAME\_OVERRUN)**  
This event indicates that a new frame has been received while the FRAME\_DONE flag was still set.
- **Data Frame Received (DATA\_FRAME)**  
This event indicates that a data frame has been received.



### 28.3.3.3 Configuring Interrupts

To configure interrupts on the FSI, the application should select the interrupt vector for each desired event using the TX\_INT\_CTRL register for the transmitter, and RX\_INT1\_CTRL and RX\_INT2\_CTRL registers for the receiver. There is no module-level interrupt enable bit to configure.

---

**NOTE:** If an event is registered for both interrupt vectors, both interrupts will fire. There are no hardware checks for overlapping interrupt vector assignments.

---

### 28.3.3.4 Handling Interrupts

Inside the interrupt service routine (ISR), the user should clear the event flag using the EVT\_CLR register and then acknowledge the CPU interrupt.

If the one event occurs multiple times before the corresponding bit is cleared by software, no new interrupt will be generated.

If multiple events occur simultaneously, or very close in time, it is possible to handle multiple conditions within a single interrupt. Each flag is independently set by hardware and must be cleared by application software. If multiple different events occur, the ISR can handle each in whatever order is deemed necessary by the application. It is not advisable to clear the full interrupt status register in every ISR. This may cause the application to miss events that may be detrimental to the application. A sample sequence for handling interrupts on the receiver follows; the transmitter routine will be similar.

- On receiving an interrupt, copy the current state of the receive event and error status flag register (RX\_EVT\_STS) into a local snapshot variable.
- Read all of the bits from the snapshot to determine the events which require action.
- Perform the necessary actions for each of the events seen in the snapshot.
- Write to the receive event and error clear register (RX\_EVT\_CLR) with the snapshot to clear only those interrupts that were set at the beginning of the ISR.
- Repeat this sequence for every generated ISR.

There is a chance that another event occurred during the just-handled ISR since only the snapshot of events was handled and then cleared; an event flag may still be set at the end of the ISR. As soon as the ISR completes, a new interrupt will be generated and this flag will still be set and can be handled accordingly.

Software accesses tied to multiple events and handled within the same ISR may cause race conditions which will cause the software to not function as desired. For example, it is recommended to use different interrupt lines if the user wants to enable events for both ping and data frames. If both are handled within the same interrupt line, the software may only respond to one of the events if they both occur close in time.

### 28.3.4 CLA Task Triggering

In addition to generating interrupt vectors to the PIE, both interrupt lines for each module TX\_INT1, TX\_INT2, RX\_INT1, and RX\_INT2 can be assigned to trigger CLA tasks. Refer to the Configuration options table for the list of all sources capable of CLA task triggering. The configuration and use of CLA tasks are described in the [Section 5.3.4](#). The CLA has access to the entire FSI register map. This allows the CLA to manage the FSI independently from the CPU, freeing it up for other tasks.

### 28.3.5 DMA Interface

Both the transmitter and receiver are capable of using the DMA for automatic data transfers. The DMA trigger is independent from the interrupt signals. DMA events are only triggered on the completion of a data frame.

The transmitter DMA trigger is enabled by setting TX\_DMA\_CTRL.DMA\_EVT\_EN to 1. The transmitter must also set TX\_OPER\_CTRL\_LO.START\_MODE to 0x2 to allow either a write to the TX\_FRAME\_CTRL.START bit or to the TX\_FRAME\_TAG\_UDATA register to start the transmission.

The receiver DMA trigger is enabled by setting RX\_DMA\_CTRL.DMA\_EVT\_EN to 1.

Refer to [Section 28.4.2](#) and [Section 28.4.3](#) for more DMA information specific to each FSI Module.

### 28.3.6 External Frame Trigger Mux

The FSI has two muxes connected to the transmitter module. These muxes are used to select triggers to start ping frames, and/or generic frames. These muxes are independently configured for each type of frame. The application may select one trigger source per frame type. Use of these triggers are optional.

The external ping frame trigger is configured by setting TX\_PING\_CTRL.EXT\_TRIG\_SEL to the index of the desired trigger. TX\_PING\_CTRL.EXT\_TRIG\_EN must also be set to allow the trigger to generate a ping frame.

The generic frame trigger is configured by setting TX\_OPER\_CTRL\_HI.EXT\_TRIG\_SEL to the index of the desired trigger. TX\_OPER\_CTRL\_LO.START\_MODE must be set to 0x1 in order for a frame to be transmitted by an external trigger.

**Table 28-3. External Trigger Sources and Their Index**

Index	External Trigger Source
0:7	Reserved
8	EPWM1-SOCA
9	EPWM1-SOCB
10	EPWM2-SOCA
11	EPWM2-SOCB
12	EPWM3-SOCA
13	EPWM3-SOCB
14	EPWM4-SOCA
15	EPWM4-SOCB
16	EPWM5-SOCA
17	EPWM5-SOCB
18	EPWM6-SOCA
19	EPWM6-SOCB
20	EPWM7-SOCA
21	EPWM7-SOCB
22	EPWM8-SOCA
23	EPWM8-SOCB
31:24	Reserved

## 28.4 FSI Operation

### 28.4.1 Introduction to Operation

The transmitter and receiver modules are two completely independent modules on the device. Each module has an independent set of control registers, clocking, and interrupts. The following sections describe the frame format and the various initialization and configuration procedures for both the transmitter and receiver.

### 28.4.2 FSI Transmitter Module

The FSI transmitter module handles the framing of data, CRC generation, and signal generation of TXCLK, TXD0, and TXD1, as well as interrupt generation. The operation of the transmitter core is controlled and configured through programmable control registers. The transmitter control registers allow the CPU (or the CLA) to program, control, and monitor the operation of the FSI receiver. The transmit data buffer is accessible by the CPU, CLA, and the DMA.

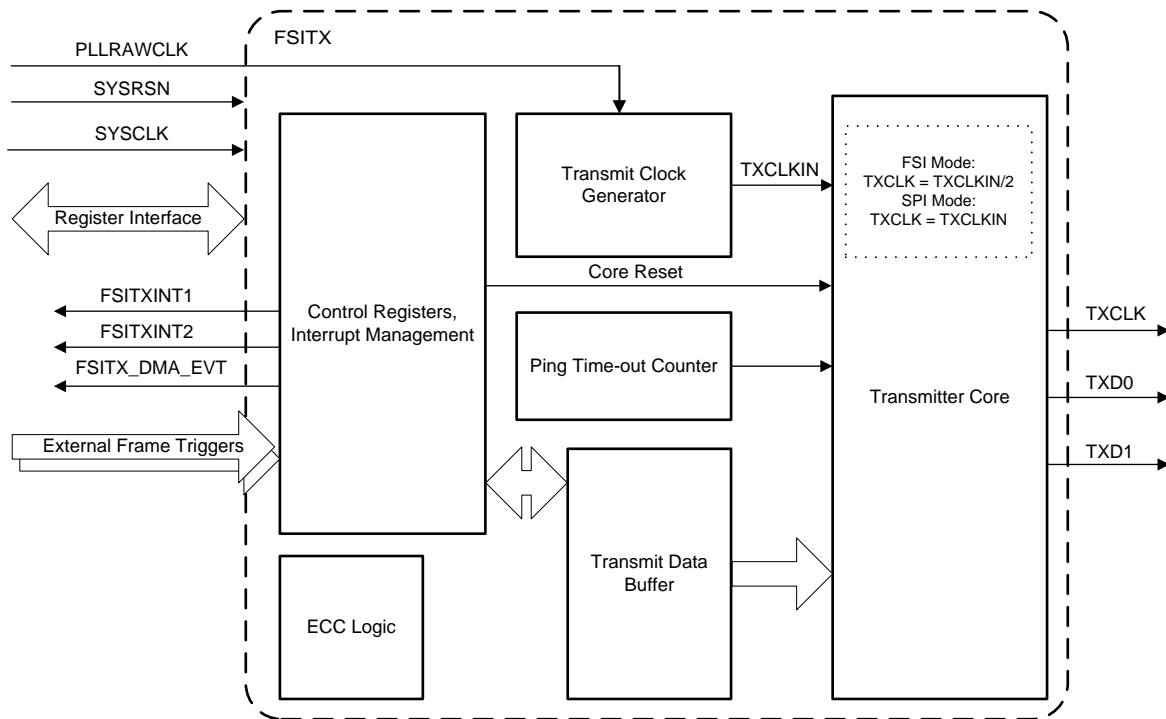
The transmitter has the following features:

- Automated ping frame generation

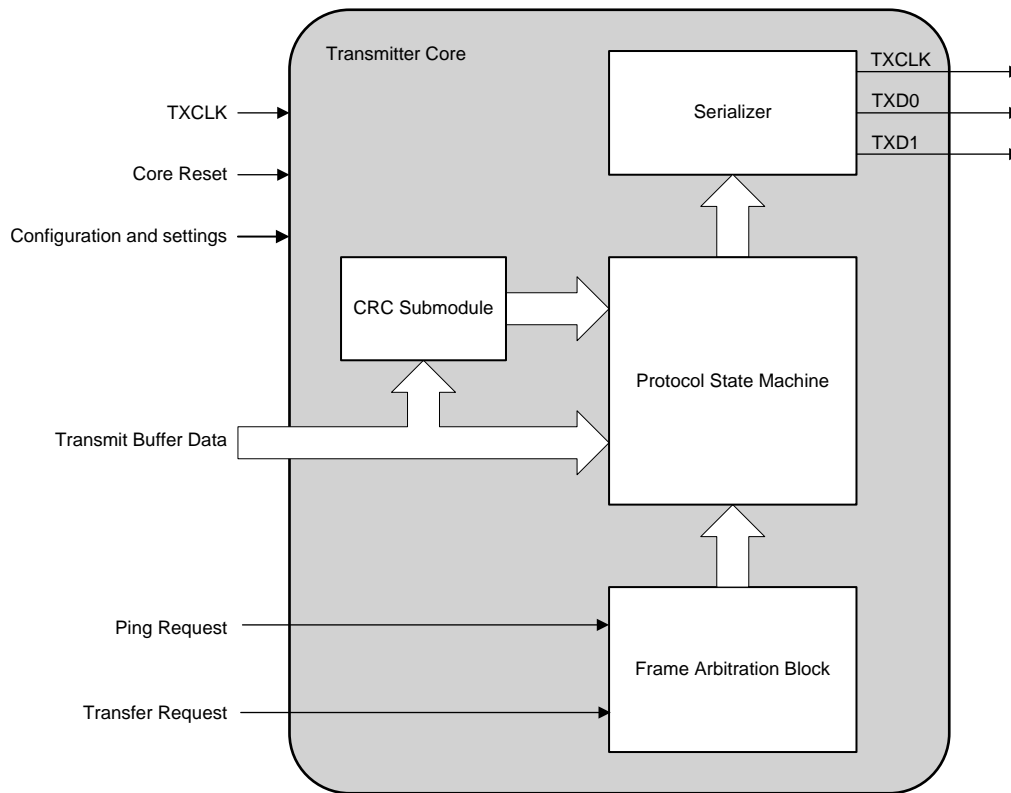
- Externally triggered ping frames
- Externally triggered data frames
- Software-configurable frame lengths
- 16-word data buffer
- Data buffer underrun and overrun detection
- Hardware-generated CRC on data bits
- Software ECC calculation on select data
- DMA support
- CLA task triggering

Figure 28-4 shows the high-level block diagram of the FSI transmitter. Figure 28-5 shows the block diagram of the transmitter core submodule.

Figure 28-4. FSI Transmitter Block Diagram



The block diagram of the transmitter core submodule is shown in Figure 28-5.

**Figure 28-5. FSI Transmitter Core Block Diagram**


The following sections describe the various aspects of the FSI transmitter in detail.

### 28.4.2.1 Initialization

On the first initialization or after a module reset due to an underrun condition, the transmitter module should execute the following initialization sequence in order to start or resume transmit operations.

1. Initialize the transmitter clock by setting TX\_CLK\_CTRL.CLK\_RST to 1 and subsequently clearing it.
2. Set the clock to the transmitter core to PLLRAWCLK by setting TX\_OPER\_CTRL\_LO.SEL\_PLLCLK to 1.
3. Set the clock prescaler value to the desired rate by writing to TX\_CLK\_CTRL.PRESCALE\_VAL.
4. Enable the transmitter clock divider by setting TX\_CLK\_CTRL.CLK\_EN to 1.
5. Assert the transmitter module soft reset by writing 0xA501 to TX\_MASTER\_CTRL.
6. Wait four TXCLK cycles.
7. Release the transmitter core from reset by writing 0xA500 to TX\_MASTER\_CTRL.

After initialization and configuration, the transmitter module should synchronize with receiver module before transmitting. The synchronization sequence is described in [Section 28.5.1](#).

#### **CAUTION**

Do not change TX\_CLK\_CTRL.PRESCALE\_VAL while the clock is enabled (TX\_CLK\_CTRL.CLK\_EN = 1). Doing so may cause undefined behavior.

### 28.4.2.2 Clocking

The transmitter core registers and control logic run off of the device system clock (SYSCLK).

The FSI Transmit Clock (TXCLK) is derived from PLLRAWCLK. PLLRAWCLK is divided down by configuring the clock prescaler value (TX\_CLK\_CTRL.PRESCALE\_VAL) then setting the clock divider enable bit (TX\_CLK\_CTRL.CLK\_EN). The clock prescaler value can be set to divide PLLRAWCLK by 1 (TX\_CLK\_CTRL.PRESCALE\_VAL = 0x0 or 0x1) through 255 (TX\_CLK\_CTRL.PRESCALE\_VAL = 0xFF). Though TXCLK and SYSCLK are both derived from PLLRAWCLK, TXCLK is asynchronous with respect to SYSCLK.

**CAUTION**

TXCLK should never be configured to be faster than SYSCLK.

### 28.4.2.3 Transmitting Frames

On the transmitter, the ping frame is the only frame which can be set up and transmitted without any further software or DMA intervention. Ping frames can be transmitted by any (or all) of the three sources: automatic ping timer, software, or external triggers.

Each available frame type can be sent multiple ways. Generically, the following steps must be executed before the frame is sent. These steps may be executed in any order before the start condition is set.

1. Configure the frame type
2. Set the frame tag
3. If the frame to be sent is a data frame:
  - Set the user data
  - Write to the data buffer
  - Set the word length if the frame is a software defined frame length
4. Set the start condition

---

**NOTE:** Note: There is no hardware check implemented to check whether the type field written by software is valid or not. If an invalid type is used and a frame transmission is initiated, the behavior will be as follows:

- The transmitted frame structure will be exactly like an NWORD data frame. The size of the data frame will be determined by the value in the TX\_FRAME\_CTRL.N\_WORDS register.
- The frame type field of the transmitted data frame will be transmitted as programmed. If this is received by an FSI receiver, it will generate a Type error.

This mechanism can be used to force a Type error in a received frame for testing purposes.

---

The following sections describe the specific configuration for each frame type and start condition.

#### 28.4.2.3.1 Software Triggered Frames

The most basic way to transmit a data frame is through software. Each step must be handled by the application. To send a data frame using software, the following steps should be executed. Steps 1-6 may be executed in any order before setting TX\_FRAME\_CTRL.START. Some fields may not need to be reconfigured for every transmission. The frame tag, user data, and frame type are sticky and will be re-transmitted in the subsequent frame unless they are modified by software.

1. Write the data to be transmitted to the next location of the transmit data buffer.
2. Set TX\_FRAME\_CTRL.FRAME\_TYPE to the appropriate value for the type of frame to be transmitted.
3. Set TX\_FRAME\_CTRL.N\_WORDS to 1 less than the number of words to be transmitted if TX\_FRAME\_CTRL.FRAME\_TYPE is set to 0011, the frame type of the software-defined length data frame. That is, if 16 words will be transmitted, N = 16, set TX\_FRAME\_CTRL.N\_WORDS to 15.
4. When the frame is assembled before transmitting, the FSITX hardware will calculate the CRC to be transmitted. If TX\_OPER\_CTRL\_LO.SW\_CRC is 1, the application may calculate a custom CRC value and then set TX\_USER\_CRC to the result.

5. Set TX\_FRAME\_TAG\_UDATA.FRAME\_TAG to the desired tag.
6. Set TX\_FRAME\_TAG\_UDATA.USER\_DATA to the desired user data.
7. Set TX\_FRAME\_CTRL.START to 1 to initiate the transmission of the data frame.

Once the frame transmission has started, the TX\_FRAME\_CTRL.START will be cleared by hardware. To monitor if the frame has completed, the software can poll TX\_EVT\_STS.FRAME\_DONE.

### **28.4.2.3.2 Externally Triggered Frames**

The transmitter can transmit frames when triggered by an external source. See [Section 28.3.6](#) for more information on the available external triggers.

To transmit frames using an external trigger, the application must follow the same procedure as described in [Section 28.4.2.3.1](#). The only difference is that in Step 7, the start condition will be automatically set when the external trigger condition is met rather than by software.

It is important to note that by externally triggering frames, the frame information to be sent will be pulled from the same registers described in the previous section. Because of this, it is possible to send any type of frame from an external trigger including ping, error, and data frames. Also, there is no hardware mechanism by which the FSI can determine if multiple triggers occur. The FSITX will take the data as is, and the application software should ensure that this data has been updated as necessary.

Using TX\_EVT\_STS fields either by polling or by interrupts, the application can populate or update the frame information to be sent in the next frame

### **28.4.2.3.3 Ping Frame Generation**

Assuming the FSI transmitter has already been properly initialized, the following sequences can be used to configure and send ping frames.

#### **28.4.2.3.3.1 Automatic Ping Frames**

To generate periodic ping frames, the following steps must be followed:

1. Initialize the ping counter by writing 1 to TX\_PING\_CTRL.CNT\_RST.
2. Set the desired ping tag to TX\_PING\_TAG.TAG.
3. Set the ping timer reference value to TX\_PING\_TO\_REF.TO\_REF.
4. Enable the ping timer by writing 1 to TX\_PING\_CTRL.TIMER\_EN.

The ping timer is a free-running counter which will count up from 0. The current value of the ping timer counter is found in TX\_PING\_TO\_CNT. When the current value of TX\_PING\_TO\_CNT matches the reference value TX\_PING\_TO\_REF.TO\_REF, TX\_EVT\_STS.PING\_TRIGGERED will be set. TX\_PING\_TO\_CNT will reset to 0 and resume counting until the next match has occurred or the ping timer is halted by software (TX\_PING\_CTRL.TIMER\_EN is set to 0).

#### **28.4.2.3.3.2 Software Triggered Ping Frame**

Software can also manually generate a ping frame. The process for sending a ping frame with software is very similar to sending the other types of frames. The following steps must be followed:

1. Set TX\_FRAME\_CTRL.FRAME\_TYPE to 0000'b to denote that the frame being sent will be a Ping Frame.
2. Set TX\_FRAME\_TAG\_UDATA.FRAME\_TAG to the desired value.
3. Write 1 to TX\_FRAME\_CTRL.START. This will start the transmission.

Once the frame transmission has started, the TX\_FRAME\_CTRL.START will be cleared by hardware. To monitor if the frame has completed, the software can poll TX\_EVT\_STS.FRAME\_DONE.

#### **28.4.2.3.3.3 Externally Triggered Ping Frame**

The last source for generating ping frames is an external trigger. One of up to 32 different triggers may be selected. See [Section 28.3.6](#) for the list of input sources.

### CAUTION

Ping frames can be triggered by both an external trigger source and the internal ping timer. If TX\_PING\_CTRL.EXT\_TRIG\_EN is set to 1, it will take precedence and the ping timer will be ignored.

#### 28.4.2.3.4 Transmitting Frames with DMA

The FSI transmitter can send data which is continuously fed with the DMA. A DMA trigger will be generated every time a data frame transmission is completed. This is concurrent with the FRAME\_DONE signal that sets the TX\_EVT\_STS.FRAME\_DONE flag.

In order to transmit continuous data with the DMA, some configurations need to be made on the transmitter:

First, set TX\_DMA\_CTRL.DMA\_EVT\_EN to 1. This will allow the DMA trigger to propagate to the DMA module. Next, TX\_OPER\_CTRL\_LO.START\_MODE must be set to 0x2. The transmitter is now able to start a transmission using a software write to TX\_FRAME\_CTRL.START or TX\_FRAME\_TAG\_UDATA..

The DMA must also be configured properly for the FSI to send the data. One possible solution of using the DMA to continuously feed the transmit buffer is shown below:

- Set up two DMA channels to be triggered by the same FSI transmitter and DMA trigger.
- Configure one channel to fill the transmit buffer.
- Configure the other channel to set the frame tag and user data fields
- Since the FSI transmit buffer is a 16-word circular buffer, ensure the DMA channel servicing the data buffer wraps the after 16 words are copied.

---

**NOTE:** Because the frame tag and user data must be written in to in order to initiate the transmission of the frame, use two consecutive DMA channels. This ensures that the DMA channels are always executed in sequence. The DMA channel servicing the data buffer should be the lower numbered channel and the tag/user data channel should be the next. For example, configure DMA channel 3 to service the data buffer, and configure DMA channel 4 to service the tag and user data.

---

#### 28.4.2.4 Transmit Buffer Management

The FSI transmitter has a 16-word buffer from which it pulls data to transmit. This buffer is implemented as a circular buffer, not a FIFO, so some care must be taken to properly interpret buffer overrun and underrun, as well as the TX\_BUF\_PTR\_STS register. These flags and pointers work under the assumption that the software or DMA is using the buffer as a circular buffer. This mode of operation is the only way in which the overrun, underrun, and pointer status are meaningful. If data is being sourced by the DMA and there is some other periodic trigger mechanism trying to initiate transfers, underrun becomes a critical error. If an underrun happens, it means that the buffer went out of sync. This would not only affect the current transfer, but all future transfers also could not be guaranteed due to the ring buffer. Under such conditions, the underrun would need a soft reset to cleanly recover. Alternately, the software could manually stop the transmitting, reset the buffer pointers, clear the remaining error conditions, and then restart transmission. The software method involves a few steps, while the soft reset is a single action and will guarantee a full reset of the control registers.

Due to the flexibility of the transmit buffer, it is possible for software to implement a simple ping-pong buffer, or to randomly load and send from any location of the buffer. If the buffer is used in this manner, error flags and status fields may be ignored without adversely affecting the transmitter capability. Additionally, the CURR\_WORD\_CNT will also be invalid if used in this way. The application can set the buffer pointer manually by writing the 4-bit index to TX\_BUF\_PTR\_LOAD. This will force the transmitter to start picking the data from the indicated location in the buffer.



### 28.4.2.5 CRC Submodule

The FSI transmitter can supply the CRC to the frame being transmitted through the embedded hardware CRC submodule or by supplying a user-defined value. This is controlled by setting TX\_OPER\_CTRL\_LO.SW\_CRC appropriately.

If hardware CRC generation is selected (TX\_OPER\_CTRL\_LO.SW\_CRC = 0, the default), the CRC is computed by hardware on the data and user data fields using the CRC polynomial  $0x7 (x^8 + x^2 + x + 1)$ . The transmitter module will automatically compute the CRC on the data fields without user intervention when the frame is transmitted. For more information on how the CRC is generated by the CRC Submodule, refer to [Section 28.4.7](#).

If software CRC generation is selected (TX\_OPER\_CTRL\_LO.SW\_CRC = 1), the CRC must be computed by software and placed in the TX\_USER\_CRC register. The next frame to be transmitted will use the value placed in the TX\_USER\_CRC register in place of the CRC value generated by the hardware.

As the TX\_USER\_CRC register is software-programmable, the application may use this field as an extra data field for application specific purposes. If TX\_USER\_CRC is used in this manner, the CRC detection on the receiver will not be valid and should be ignored.

### 28.4.2.6 Conditions in Which the Transmitter Must Undergo a Soft Reset

Unlike the receiver, there are no detectable errors that require a soft reset. A buffer overrun or underrun interrupt may or may not require a soft reset in order to resume proper operation. This determination is up to the application software. Refer to [Section 28.4.2.4](#) for more information on the transmit buffer.

### 28.4.2.7 Reset

The entire transmitter module and all transmitter registers are reset by SYSRSn. The transmitter core is reset by SYSRSn or by writing a 1 to TX\_MASTER\_CTRL.CORE\_RST.

A module reset will cause the registers to be reset to their default state.

## 28.4.3 FSI Receiver Module

The receiver module interfaces to the FSI clock (RXCLK), and data lines (RXD0 and RXD1) after they pass through an optional programmable delay line. The receiver core handles the data framing, CRC computation, and frame-related error checking. The receiver bit clock and state machine are run by the RXCLK input, which is asynchronous to the device system clock.

The receiver control registers allow the CPU (or the CLA) to program, control, and monitor the operation of the FSI receiver. The receive data buffer is accessible by the CPU, CLA, and the DMA.

The receiver core has the following features:

- 16-word data buffer
- Multiple supported frame types
- Ping frame watchdog
- Frame watchdog
- CRC calculation and comparison in hardware
- ECC detection
- Programmable delay line control on incoming signals
- DMA support
- CLA task triggering
- SPI compatibility mode

[Figure 28-6](#) provides a high-level overview of the internal modules present in the FSI receiver. [Figure 28-7](#) shows a view of the FSI receiver core submodule. Not all data paths and internal connections are shown.



Figure 28-6. FSI Receiver Block Diagram

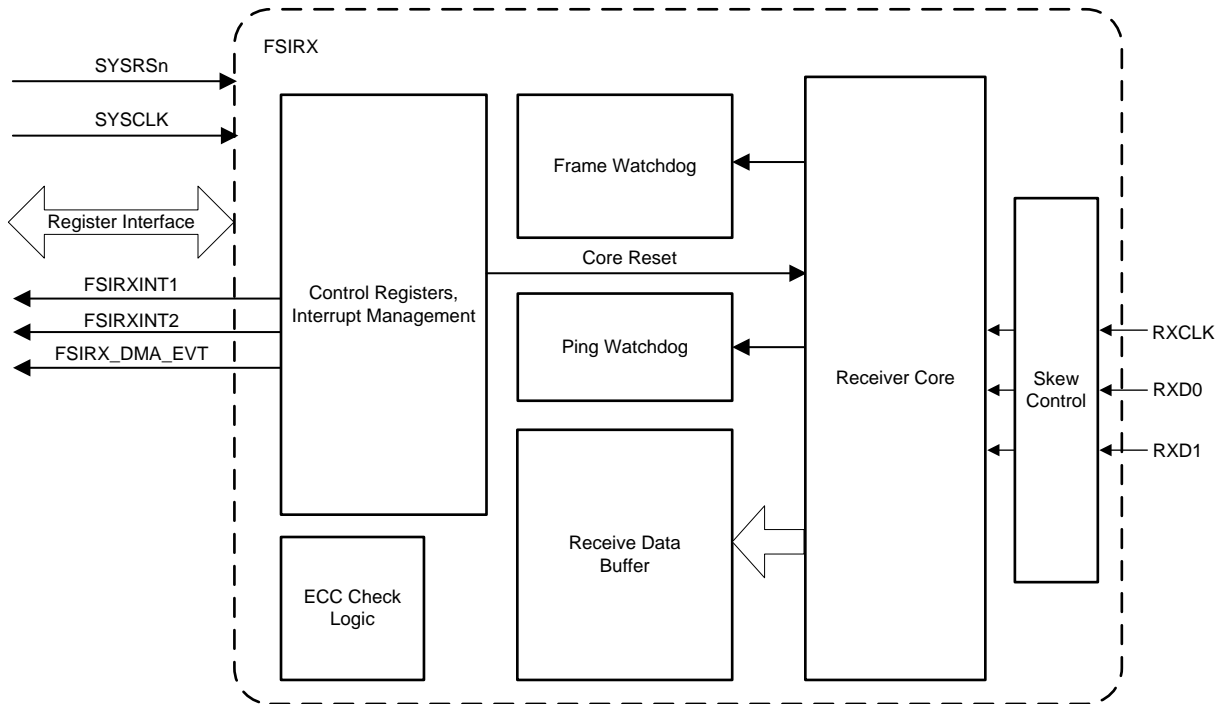
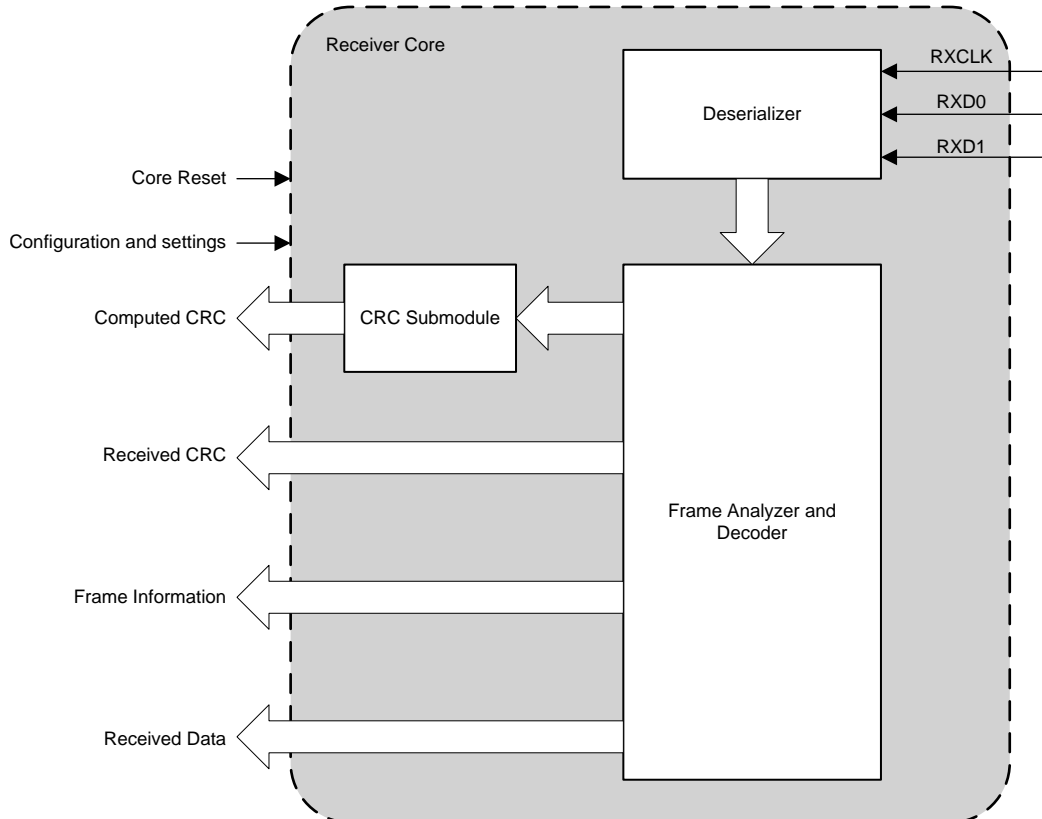


Figure 28-7. FSI Receiver Core Block Diagram



The following sections describe the various aspects of the FSI receiver module

### 28.4.3.1 Initialization

On the first initialization or after a module reset following any frame error, the receiver module should assert and release the receiver core reset bit (RX\_MASTER\_CTRL.CORE\_RST) prior to any other initialization. Once the receiver module is initialized, the following steps may be executed:

1. If required, assign interrupt sources to the necessary interrupt line.
2. If required, configure the ping watchdog to periodically check for an active link to the transmitter. See [Section 28.4.3.4](#) for configuration details.
3. If required, configure the frame watchdog to ensure that each frame is received within a predetermined window. See [Section 28.4.3.5](#) for configuration details.
4. Initialize the receive buffer pointer by writing to the RX\_BUF\_PTR\_LOAD register. Received data will be placed into the buffer starting with the address loaded in this register.
5. Ensure all errors and flags have been cleared from the RX\_EVT\_STS register.

At this point the receiver is ready to receive any incoming frames. Software can now either poll on the RX\_EVT\_STS register for various conditions. For example, when the RX\_EVT\_STS.FRAME\_DONE and no other flags are set, the receiver has successfully received a frame without error.

Next, the application should configure the various features such as the ping and frame watchdogs, DMA, external triggering, and so on. These features are described in subsequent sections. The receiver module is now ready to synchronize with the transmitter then begin reception. The synchronization sequence is described in [Section 28.5.1](#).

### 28.4.3.2 Clocking

The receiver module registers and control logic are clocked by the device system clock (SYSCLK). The receiver state machine is clocked by the receiver input clock pin (RXCLK).

#### **CAUTION**

RXCLK should never be faster than SYSCLK.

### 28.4.3.3 Receiving Frames

Once the receiver has been properly configured and synchronized, incoming messages are handled as described below. It is important to note that there is no equivalent to a chip select signal to gate incoming data. Every valid clock edge will latch data into the receiver.

The header information of the received frame will be placed in their respective register fields.

- RX\_FRAME\_INFO.FRAME\_TYPE will contain the received frame type.
- RX\_FRAME\_TAG\_UDATA.FRAME\_TAG will contain the received frame tag.
- RX\_FRAME\_TAG\_UDATA.USER\_DATA will contain the received user data.

If any error conditions occur during reception such as a CRC mismatch, frame error, frame timeout, buffer overrun, or ping watchdog timeout, the corresponding flag will be set in the RX\_EVT\_STS register.

---

**NOTE:** If at any point during operation a frame error occurs, the receiver module must be reset and re-synchronized with the transmitter before the next frame can be successfully received. The follow errors are classified as frame errors:

- Type error
  - CRC error
  - End of frame error
-

### 28.4.3.3.1 Receiving Frames with DMA

The FSI receiver can continuously receive data and move it from the receiver buffer with the DMA. A DMA trigger will be generated every time a data frame has been received. This is concurrent with the FRAME\_DONE signal that sets the RX\_EVT\_STS.FRAME\_DONE flag. In order to receive continuous data with the DMA, some configurations need to be made on the receiver.

First, set RX\_DMA\_CTRL.DMA\_EVT\_EN to 1. This will allow the DMA trigger to propagate to the DMA module. The receiver is now able to trigger a DMA event upon the reception of a data frame.

The DMA must also be configured properly for the FSI to receive the data. One possible solution for using the receiver to continuously feed the DMA is show below:

- Set up two DMA channels to be triggered by the FSI Receiver DMA Trigger.
- Configure one DMA channel to copy data from the receive buffer to a larger data buffer.
- Configure the next DMA channel to copy the received frame tag and user data to another data buffer.
- Since the FSI receive buffer is a 16-word circular buffer, ensure the DMA channel servicing the data buffer wraps after 16 words are copied.

Unlike the transmitter, there is no requirement to have the DMA channel which is handling the data buffer, execute before the DMA channel handling the received tag and user data.

### 28.4.3.4 Ping Frame Watchdog

The ping frame watchdog is a hardware-enabled automatic error detection of the connection status to the transmitter. This watchdog monitors the time elapsed between ping frames. If the transmitter has been set up to periodically send out a ping frame, the receiver can be set up to monitor whether this frame has been received within a specified amount of time. If the time between ping frames has exceeded the programmed number of clock cycles, an event will be triggered which can generate an interrupt or be monitored by software.

This watchdog has a dedicated counter which is reset and restarted upon the successful reception of a ping frame. The watchdog counter will be incremented at the rate of SYSCLK. Optionally, the watchdog can be configured to be reset upon the successful reception of any frame. This option allows the receiver to monitor for any successful frame to indicate that the connection is still alive and the transmitter is still functioning as expected.

To configure the ping frame watchdog for operation:

1. Reset the ping watchdog counter by setting RX\_PING\_WD\_CTRL.PING\_WD\_RST to 1 and then subsequently clearing it to 0.
2. Set RX\_OPER\_CTRL.PING\_WD\_RST\_MODE to the desired watchdog reset event, either 0 for ping frames only, or 1 for any frame.
3. Set RX\_PING\_WD\_REF to the maximum time between frames. Add 10 additional SYSCLK cycles to account for clock synchronization.
4. Enable the ping watchdog by setting RX\_PING\_WD\_CTRL.PING\_EN to 1.

The ping watchdog is now enabled and can now monitor for ping frames.

If the RX\_PING\_WD\_CNT value reaches the value programmed in RX\_PING\_WD\_REF, the RX\_EVT\_STS.PING\_WD\_TO flag will be set. If configured, an interrupt can be generated on this event.

### 28.4.3.5 Frame Watchdog

The frame watchdog is an additional feature the receiver can use to monitor for any error conditions. This dedicated watchdog monitors the duration that it takes for a single frame to be received. The watchdog starts incrementing at the time the receiver detects a proper start of frame condition. If the end of frame condition is not detected within the expected number of SYSCLK cycles, the frame watchdog will be triggered which can generate an interrupt or be monitored by software.

This watchdog is automatically started and stopped at the start-of-frame and end-of-frame conditions respectively. The frame watchdog is connected to SYSCLK.

To configure the frame watchdog for operation:

1. Reset the frame watchdog counter by setting `RX_FRAME_WD_CTRL.FRAME_WD_CNT_RST` to 1 and then subsequently clearing it to 0.
2. Set `RX_FRAME_WD_REF.FRAME_WD_REF` to the maximum number of `SYSCLK` cycles expected to be in the longest frame that may be received. Add an additional 10 `SYSCLK` cycles to account for clock synchronization.
3. Enable the frame watchdog by setting `RX_FRAME_WD_CTRL.FRAME_WD_CNT_EN` to 1.

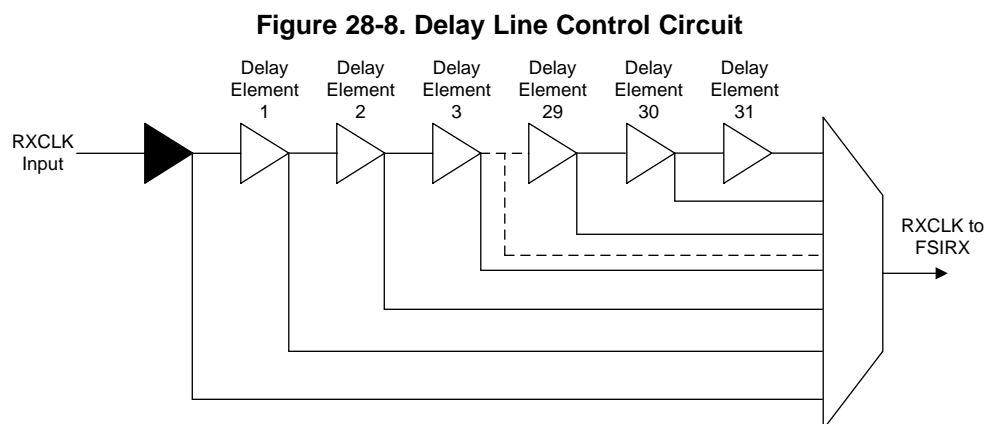
The frame watchdog is now enabled and can detect a failed frame.

If the `RX_FRAME_WD_CNT` reaches the value programmed in `RX_FRAME_WD_REF`, the `RX_EVT_STS.FRAME_WD_TO` flag will be set. If enabled, an interrupt can be generated on this event.

If the frame watchdog interrupt ever occurs, the receiver core is in an invalid state to receive a new transmission. The only way to recover from a frame watchdog time out is to undergo a soft reset, and subsequently resynchronizing with the transmitter.

### 28.4.3.6 Delay Line Control

The receiver module has a programmable delay line on each of the external signal inputs: `RXCLK`, `RXD0`, and `RXD1`. The delay elements introduce delays on the respective lines. This is to facilitate adjustment for signal delays introduced by system level components such as signal buffers, ferrite beads, isolators, and so on, or board delays such as uneven trace lengths, long cable length, and so on. The length of the delay is controlled by setting the `RX_DLY_LINE_CTRL` register values for each line. By default, no delay is introduced by the delay line elements. The delay values should only be adjusted while the `FSIRX` is held in soft reset, ensuring that there are no active transmissions during this process. Figure 28-8 shows a representation of the delay line circuitry for the input signals. The implementation for `RXCLK`, `RXD0`, and `RXD1` are replicas of this diagram. All circuits will behave similarly.



For more information on skew compensation, please refer to <http://www.ti.com/lit/an/spracj9/spracj9.pdf>.

### 28.4.3.7 Buffer Management

The FSI receiver has a 16-word buffer into which the data is copied to when it has been received. This buffer is implemented as a circular buffer, not a FIFO, so some care must be taken to properly interpret buffer overrun and underrun as well as the `RX_BUF_PTR_STS` register. These flags and pointers work under the assumption that the software or DMA is using the buffer as a circular buffer. If the receiver state machine enters into an erroneous state, there is no way for software to cleanly handle this because there is no guaranteed receive clock. For the receiver to detect a clean resynchronization, the state machine needs to be operational and not in the error state. The only way to recover from the error state is to reset the entire receiver module. For overrun and underrun, the receiver can no longer guarantee which values in the buffer are valid. As such, the best way to recover is to reset the FSI and resynchronize with the transmitter.

Due to the flexibility of the receive buffer, it is possible for software to implement a simple ping-pong buffer, or to randomly receive and read from any location of the buffer. If the buffer is used in this manner, these flags and status fields may be ignored without adversely affecting the receiver capability. Additionally, the `CURR_WORD_CNT` will also be invalid if used in this way. The application can set the buffer pointer manually by writing the 4-bit index to `RX_BUF_PTR_LOAD`. This will force the receiver to start storing the received data starting at the indicated location in the buffer.

#### 28.4.3.8 CRC Submodule

The receive module will automatically calculate the CRC on the incoming data. The received CRC value is placed into `RX_CRC_INFO.RX_CRC`. The CRC value calculated by hardware on the received data is placed into `RX_CRC_INFO.CALC_CRC`. These values are compared by hardware and will set `RX_EVT_STS.CRC_ERROR` if there is a mismatch. The receiver can generate an interrupt based on `RX_EVT_STS.CRC_ERROR` if enabled.

Since the CRC is only used in data frames, the values found in `RX_CRC_INFO.RX_CRC` and `RX_CRC_INFO.CALC_CRC` are undefined during ping and error frames.

For more information on how the CRC is calculated, refer to [Section 28.4.7](#).

If the transmitting module is sending a software-defined CRC value (`FSITX.TX_OPER_CTRL_LO.SW_CRC = 1`) the receiver module will trigger a CRC Error event if the received value does not match the hardware-calculated value. As this is an application level decision, the FSIRX can safely disregard the CRC error event. Application software will need to calculate and verify the incoming CRC using the same custom algorithm used on the transmitter and act appropriately.

The CRC field may have also been used as an application-specific value, not a CRC, the application can use the `RX_CRC_INFO.RX_CRC` as required. All CRC errors and flags can be ignored in this situation.

#### 28.4.3.9 Using the Zero Bits of the Receiver Tag Registers

The receiver tag registers (receiver frame tag and user data (`RX_FRAME_TAG_UDATA`) register and receiver ping tag (`RX_PING_TAG`) register) have the Least Significant Bit is set to 0. The actual received tag is in the bit positions 4:1. The reason for this is to facilitate user software to create a table of functions which can be called depending on the tag value. A function pointer needs a 32-bit storage space and hence each successive pointer will be offset by 2. If the first pointer is at address  $x$ , then the second pointer will be at address  $x+2$ , the third at address  $x+4$  and so on. By keeping the LSB 0, the five bits of the tag register (bits 4:0) can now be directly used as an index into a table of function pointers.

#### 28.4.3.10 Conditions in Which the Receiver Must Undergo a Soft Reset

The receiver will receive data on every clock edge. While there are specific patterns which determine the start of a frame, and denote the end of a frame, these patterns are able to occur at any point during normal operation inside of the frame. If there ever is a point at which the receiver fails to detect a successful frame, the module must be reset in order to ensure that subsequent frames are received properly.

When any of the following errors occur in a received frame, the receiver may be required to be reset and resynchronized with the transmitter:

- Frame type error
- End of frame error
- Ping frame watchdog timeout
- Frame watchdog timeout
- Receiver in an invalid state due to noisy clock

The receiver core status (`RX_VIS_1.RX_CORE_STS`) can be monitored to determine if it has entered into an error state requiring a soft reset in order to resume communication. Incorrect frame type and end of frame errors will always cause this bit to become set. A soft reset is required in these cases. A frame watchdog timeout will always require a reset due to the fact that the receiver state machine is still expecting more information when the watchdog timed out. `RX_CORE_STS` can be used to determine if a noise event was the cause of the failed frame. The ping frame watchdog also will not cause

RX\_CORE\_STS to be set. Similar to the frame watchdog, a corrupt receiver may not be the reason for the ping frame to have timed out. The transmitter may have gone offline and never sent a ping frame. Alternately, during idle time, a noise event may have occurred, thereby putting the receiver into a corrupt state. As the receiver will be able to detect this during the ping frame watchdog timeout interrupt handler, this type of event will not be lost and the application can act appropriately.

As the receiver is clocked by RXCLK, not SYSCLK, a noisy clock or data line may cause some internal design constraints to be violated, putting the receiver core logic into undefined states. Ensure that the clock and data lines satisfy the Electrical Characteristics and timing requirements of the FSI module found in the data manual for this device. Failure to do so may cause the receiver state machine to go into an unrecoverable error state. The receiver can only be recovered by undergoing a soft reset. To determine the state of the receiver core after an unexpected frame error, the application should check the receiver core status bit.

In addition to the above errors, buffer overrun or underrun may warrant a soft reset in order to resynchronize with the local application software. Refer to [Section 28.4.3.8](#) for more information on the receive buffers. The requirement of resetting the receiver due to overrun or underrun is up to the application.

After the receiver has been placed into soft reset, the application must notify the other device's transmitter to begin a new synchronization phase. The simplest way to achieve this is through a ping or error frame sent with a designated tag. If the application is not using the FSITX on the device with the detected error, some other method must be established. The other device should stop transmitting and begin a new synchronization phase.

#### 28.4.3.11 Reset

The receiver module and its registers are reset by SYSRSn. The receiver core is reset by SYSRSn or by writing a 1 to RX\_MASTER\_CTRL.CORE\_RST.

A module reset will cause the registers to be reset to their default state. After a module reset, the receiver module must be re-initialized, and the data link re-established.

### 28.4.4 Frame Format

The FSI module transmits and receives information in frames. Each frame will contain multiple phases where different information can be found. The number of phases as well as the total length of the frame will vary depending on the frame type being transmitted. Frames may be as short as 16 bits long for a ping or error frame or 288 bits long for a 16-word data frame.

In normal transmission mode, there are four preamble clock edges before the start of the frame and four post-frame clock edges (postamble). Data is transmitted on both edges of the clock (double data rate). The basic frame structure is shown below. Each phase of the frame (such as start-of-frame, frame type, and so on) will be transmitted with the most significant bit first. [Table 28-4](#) describes the basic frame structure used by the FSI and adapted according to which frame type is transmitted.

**Table 28-4. Basic Frame Structure**

Idle state	Preamble	Start of Frame	Frame Type	User Data	Data Words	CRC byte	Frame Tag	End of Frame	Postamble	Idle state
	1111	1001	4 bits	8 bits	1-16 words	8 bits	4 bits	0110	1111	

The FSI also supports a SPI compatibility mode. The SPI compatible frame structure is similar to a standard FSI frame, but there are differences. Refer to [Section 28.4.9](#) for more information on how to configure and use the SPI compatibility mode.

---

**NOTE:** One word of the FSI refers to 16 bits.

The terms “frame” and “packet” can be used interchangeably to describe the signaling format of the FSI .

---



### 28.4.4.1 FSI Frame Phases

The different phases of the frame structure are described in detail below:

- **Idle State**  
During the idle state, the clock and data lines will be driven high, the inactive state.
- **Preamble**  
The preamble phase contains four clock edges (or two complete clock pulses) with the data signals held in the high state. These clock edges serve to flush the receiver logic and prepare it for receiving a new frame. This phase is not present in SPI compatibility mode.
- **Start of Frame**  
The start of frame phase contains two clock pulses with four bits, 1001, transmitted on the data lines.
- **Frame Type**  
The frame type phase contains two clock pulses with the 4-bit frame type code being transmitted on the data lines. The different frame types are described in detail in [Section 28.4.4.2](#). The transmitter must set the TX\_FRAME\_CTRL.FRAME\_TYPE field before transmitting a frame. The received frame type is stored in the RX\_FRAME\_INFO.FRAME\_TYPE.
- **User Data**  
The user data phase contains a fully user-configurable data field. There are no restrictions on how this field is used. This phase is only available in data frames. The user data to be transmitted is set by writing to TX\_FRAME\_TAG\_UDATA.USER\_DATA. The received user data is stored in RX\_FRAME\_TAG\_UDATA.USER\_DATA.
- **Data**  
The data phase contains the data that is being transmitted. The data is pulled from the transmit buffer of the transmitter and will be placed in the receive buffer of the receiver. Word 0 is transmitted first. This phase is only present in data frames. Depending on the type of frame transmitted, this can contain anywhere between 1 and 16 words depending on the frame type selected. More information on data frames can be found in [Section 28.4.4.2.3](#).
- **CRC Byte**  
The CRC byte contains the CRC of the transmitted data. The value present in this phase can be sourced from either hardware or software based on the TX\_OPER\_CTRL\_LO.SW\_CRC bit. Refer to the module-specific section of the CRC Submodule for more information on the CRC is generated or used, for the transmitter and receiver modules respectively. The CRC byte is only present in data frames.
- **Frame Tag**  
The frame tag contains the 4-bit user-defined frame tag. There are no restrictions on how this field is used in an application. The transmitter supplies this tag into the TX\_FRAME\_TAG\_UDATA.FRAME\_TAG bits for data frames. Ping frames use the tag defined in TX\_PING\_TAG.TAG. The receiver can access the received frame tag in RX\_FRAME\_TAG\_UDATA.FRAME\_TAG.
- **End of Frame**  
The end of frame contains four clock edges with four bits, 0110, transmitted on the data lines.
- **Postamble**  
The postamble contains four additional clock edges with the data lines held in the high state. After the postamble, the clock and data lines will be driven high, their inactive state. This phase is not present in SPI compatibility mode.

### 28.4.4.2 Frame Types

The FSI hardware can generate and handle many predefined frame types. The different frame types can be used by the application to signal different types of events or convey different information to the receiver. The different frame types influence which phases and data fields to include in the transmitted frames.

[Table 28-5](#) provides a short overview of the different frame types used by the FSI. Each frame type is described in more detail in the following subsections.

**Table 28-5. Frame Types and their 4-bit codes**

Frame Type	4-bit Frame Code	Description
PING	0000	This is the ping frame which can be sent either by software or automatically by hardware.
ERROR	1111	This should be used typically during error conditions or any condition where one side wants to signal the other side for attention. However, the user software is at liberty to use this for any purpose.
DATA_1_WORD	0100	1 word data packet (16 bits of data)
DATA_2_WORD	0101	2 word data packet (32 bits of data)
DATA_4_WORD	0110	4 word data packet (64 bits of data)
DATA_6_WORD	0111	6 word data packet (96 bits of data)
DATA_N_WORD	0011	N(1-16) word data packet where software has programmed the number of the data words in a designated register. Both transmitter and receiver modules should have the same value programmed.
Reserved	0001, 0010, and 1000-1110	Reserved

#### 28.4.4.2.1 Ping Frames

Ping frames are one of the most basic frames that can be generated by the FSI. [Table 28-6](#) shows the structure of the ping frames.

**Table 28-6. Ping Frame**

Idle state	Preamble	SOF	Frame Type	Frame Tag	EOF	Postamble	Idle state
	1111	1001	0000	xxxx	0110	1111	

The ping frame type is always 0000. The frame tag is defined by the application. Separate frame tags exist for timer and software initiated ping frames. No data or CRC is transmitted in a ping frame.

The main purpose of the ping frame is to periodically send a notification to the receiver to ensure an active connection between the transmitter and receiver. The transmitter and receiver cores implement different features to allow the ping frame to operate as a line break detect feature.

On the transmitter, the ping frame is the only frame which can be set up and transmitted without any further software or DMA intervention. Ping frames can be transmitted by any (or all) of the three sources: automatic ping timer, software, or external triggers. See [Section 28.4.2.3.3](#) for information on how the transmitter configures and sends the ping frames.

The receiver has a ping watchdog that can detect if a ping frame has not been received in a predetermined window. This allows the receiver to know if the connection between it and the transmitter has been broken. See [Section 28.4.3.4](#) for information on how the receiver handles ping frames.

#### 28.4.4.2.2 Error Frames

Error frames are similar to ping frames in that there are no data fields transmitted. Despite the naming of this frame as an “error frame,” the usage of it is up to the application, as no restrictions are placed on how and when this type of frame is transmitted. [Table 28-7](#) shows the structure of an error frame.

**Table 28-7. Error Frame**

Idle state	Preamble	SOF	Frame Type	Frame Tag	EOF	Postamble	Idle state
	1111	1001	1111	xxxx	0110	1111	



The structure of the error frame is the same as a ping frame. No data or CRC values are transmitted. The frame type is 1111 for all error frames, and the frame tag is defined by software in the TX\_FRAME\_TAG\_UDATA register.

The receiver can detect if an error frame has been received based on the frame type field. Because of this, the receiver can read the incoming frame tag from the RX\_FRAME\_TAG\_UDATA register and act on up to 16 different conditions.

#### 28.4.4.2.3 Data Frames

Data frames are the most complex frames. As the name indicates, these frames are used to transfer data. Table 28-8 shows the general structure of data frames.

**Table 28-8. Data Frame**

Idle state	Preamble	SOF	Frame Type	User Data	Data Words	CRC Byte	Frame Tag	EOF	Postamble	Idle state
	1111	1001	0xxx	xxxx xxxx	1-16 words	xxxx xxxx	xxxx	0110	1111	

The frame type field will reflect the 4-bit code of the frame type. A list of frame types can be seen in Table 28-5. The number of the data words transmitted will be determined by the frame type chosen.

There are four fixed-length data frames supported by the frame type: 1 word, 2 words, 4 words, and 6 words.

Additionally, there is a user-defined data length frame type where the number of data words is fixed by software. Anywhere from 1 to 16 words can be transmitted in this frame type. This length must be configured in the N\_WORDS field of the transmitter's TX\_FRAME\_CTRL register and receiver's RX\_OPER\_CTRL register.

#### 28.4.4.3 Multi-Lane Transmission

The FSI is capable of transmitting and receiving data on two parallel data lines. When enabled, data bits will be split between the data lines while the start of frame, frame type, frame tag, and end of frame fields will be identical and complete on each line. The user data, data, and CRC fields will be split between the data lines. Starting with the most significant bit, the odd-numbered bits appear on D0 and even-numbered bits appear on D1.

In the example below, assume the following:

8-bit user data: u7u6u5u4u3u2u1u0

16-bit data: d15d14d13d12...d1d0

8-bit CRC: c7c6c5c4c3c2c1c0

**Table 28-9. Multi-Lane Frame Format**

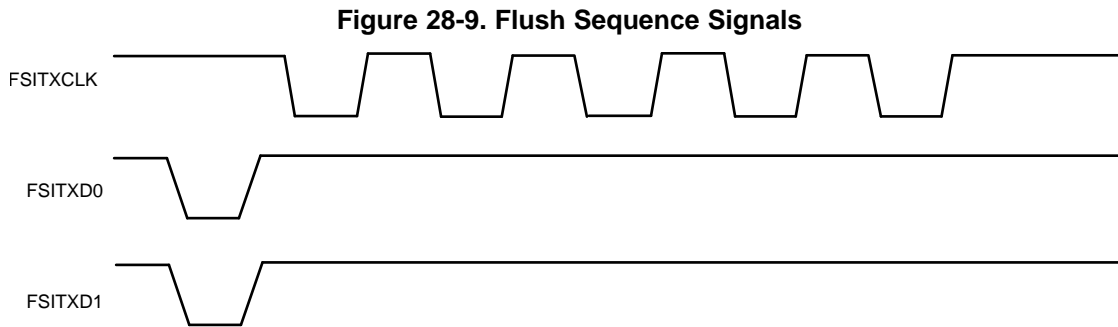
Idle state	Preamble	SOF	Frame Type	User Data	Data Words	CRC Byte	Frame Tag	EOF	Postamble	Idle state
TXD0	1111	1001	0011	u <sup>7</sup> u <sup>5</sup> u <sup>3</sup> u <sup>1</sup>	d <sup>15</sup> d <sup>13</sup> ...d <sup>1</sup>	c <sup>7</sup> c <sup>5</sup> c <sup>3</sup> c <sup>1</sup>	xxxx	0110	1111	
TXD1	1111	1001	0011	u <sup>6</sup> u <sup>4</sup> u <sup>2</sup> u <sup>0</sup>	d <sup>14</sup> d <sup>12</sup> ...d <sup>0</sup>	c <sup>6</sup> c <sup>4</sup> c <sup>2</sup> c <sup>0</sup>	xxxx	0110	1111	

### 28.4.5 The Flush Sequence

Every time there is a soft reset of the receiver, the receiver requires a flush sequence from the transmitter before it can receive and decode frames. The receiver core has an asynchronous reset mechanism which will allow the receive module to be reset even in the absence of the receive clocks. However, due to the design, this reset will be released synchronous to the receive clock (RXCLK). Thus, the receiver will require five full clock pulses to be able to come out of reset. Sending the flush pattern will ensure that these clock edges are received and any subsequent frames sent to the receiver will be correctly interpreted.

The flush sequence consists of a single toggle on both of the data lines as well as five consecutive pulses on the clock line.

Figure 28-9 shows a sample plot of the flush sequence.



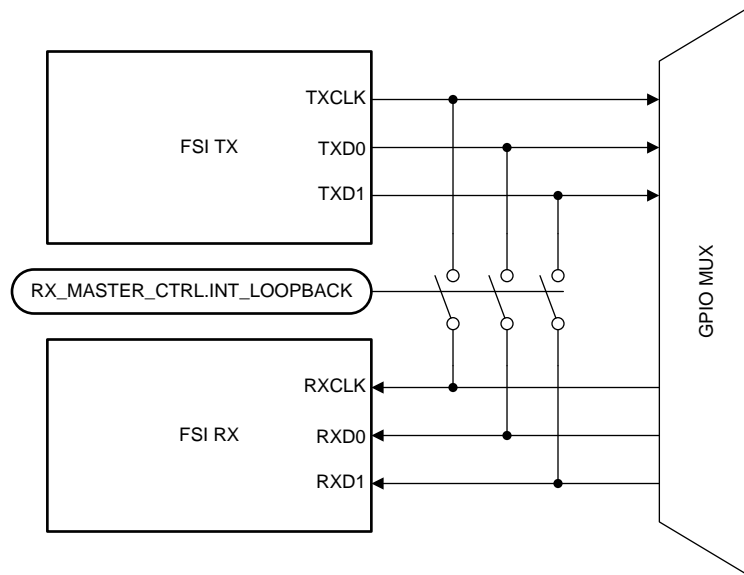
If the FSI receiver is receiving data from a standard SPI, a data word of 0xFFFF from the SPI will have the same effect as a flush sequence.

### 28.4.6 Internal Loopback

The transmitter and receiver cores can be connected together internally to allow for development and debug. This is achieved by setting `RX_MASTER_CTRL.INT_LOOPBACK` to 1. Internal loopback will route the signals from the corresponding transmitter to the appropriate receiver pin. No configuration needs to be done in the transmitter.

Figure 28-10 shows the signal connections with internal loopback.

Figure 28-10. FSI with Internal Loopback



### 28.4.7 CRC Generation

The FSI uses CRC-8 with the polynomial 0x07 for the internal hardware CRC generation. This polynomial is also represented as  $x^8+x^2+x+1$ .

For example, for a 2-word data packet the following calculation would occur:

Data-1 = 0x4433

Data-0 = 0x2211

User Data = 0xAA

The CRC would be computed with the bytes being taken in the following order (first to last)

0xAA – Byte 0, User Data

0x11 – Byte 1, Data-0, Least significant byte

0x22 – Byte 2, Data-0, Most significant byte

0x33 – Byte 3, Data-1, Least significant byte

0x44 – Byte 4, Data-1, Most significant byte

### 28.4.8 ECC Module

The FSI module comes with a 16-bit or 32-bit ECC computation module in both the transmitter and receiver. Use of this module is optional.

Note that the ECC is independent and unrelated to the hardware CRC computation module present in both the transmitter and receiver cores.

The following example shows a scenario in which the application requires ECC be calculated and transmitted on a 2-word data frame.

In the FSITX module:

1. Configure the ECC module for 32-bit data by setting TX\_OPER\_CTRL\_HI.ECC\_SEL to 1.
2. Write the data to the TX\_ECC\_DATA register as well as the transmit buffer.
3. Read TX\_ECC\_VAL Register. This register contains the 8-bit ECC value calculated on the data.
4. Copy the 8-bit data from TX\_ECC\_VAL to TX\_FRAME\_TAG\_UDATA.USER\_DATA.
5. Set the Start Condition to begin the transmission.

The reverse process is followed on the FSIRX module. Once the data frame is received, user software can do the following:

1. Copy the data from the receive buffer to the RX\_ECC\_DATA register.
2. Copy the received user data that contains the transmitted ECC value from RX\_FRAME\_TAG\_UDATA.USER\_DATA to the RX\_ECC\_VAL register.
3. Read the RX\_ECC\_LOG register. This contains the result of the ECC computation using the RX\_ECC\_DATA and RX\_ECC\_VAL registers.
  1. If no ECC errors were detected, RX\_ECC\_LOG will be 0. The correct data will be available in RX\_ECC\_SEC\_DATA.
  2. If a single bit error was detected, RX\_ECC\_LOG.SBE will be 1. The autocorrected data will be available in RX\_ECC\_SEC\_DATA.
  3. If multiple bit errors occurred, RX\_ECC\_LOG.MBE will be set to 1. The data in RX\_ECC\_SEC\_DATA is invalid and should not be used.

Using a 2-word data frame plus using the user data for the ECC is one possible implementation for ECC detection. Another option is to use a larger data frame and allocate one of the data words to be the ECC value.

### 28.4.9 SPI Compatibility Mode

The FSI supports a SPI compatibility mode. While the FSI can communicate with a standard SPI module, the FSI supports a limited configuration. The features of this compatibility mode follow:

- Data will transmit on rising edge and receive on falling edge of the clock.
- Only 16-bit word size is supported.
- TXD1 will be driven like an active-low, chip-select signal. The signal will be low for the duration for the full frame transmission.
- No receiver chip-select input is required. RXD1 is not used. Data is shifted into the receiver on every active clock edge.
- No preamble or postamble clocks will be transmitted. All signals return to the IDLE state after the frame phase is finished.

- It is not possible to transmit in the SPI slave configuration because the FSI TXCLK cannot take an external clock source.

Table 28-10 lists the frame structure of the SPI compatibility mode. Each frame phase is present in this mode. If the FSI is transmitting to a standard SPI module, the SPI must decode the frame structure. Similarly, if the FSI is configured as a SPI slave, the standard SPI must encode the transmission to be sent.

**Table 28-10. SPI Compatibility Frame Structure**

Idle state	Start of Frame	Frame Type	User Data	Data Words	CRC byte <sup>(1)</sup>	Frame Tag	End of Frame	Idle state
	1001	4 bits	8 bits	1-16 words	8 bits	4 bits	0110	

<sup>(1)</sup> The CRC byte is present only in data frames.

Because of the requirement that the standard SPI module encodes the various frame data, this limits the type of modules that can be connected to the FSI in SPI mode. The paired SPI module must have enough functionality to encode and decode the frames.

If the FSI is transmitted to a standard 16-bit SPI, the data would be arranged in the following manner. The example provided in Table 28-11 assumes a DATA\_2\_WORD frame has been sent.

**Table 28-11. Contents of Data Received by a Standard SPI**

SPI Data	Data Contents
SPI word 0	1001, 0100, 8-bit User Data
SPI word 1	Data word 1
SPI word 2	Data word 2
SPI word 3	8-bit CRC, 4-bit Frame Tag, 0110

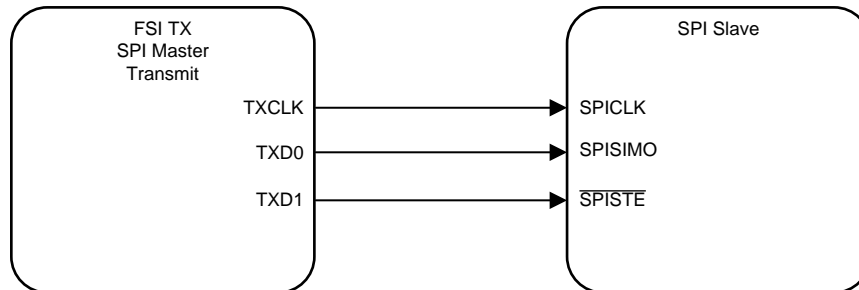
### 28.4.9.1 Available SPI Modes

There are a few wiring schemes available for the FSI to use when communicating with a SPI module.

#### 28.4.9.1.1 FSITX as SPI Master, Transmit Only

The FSITX can operate as an independent SPI Master module. In this condition, TXCLK will be connected to SPICLK, TXD0 will be connected to SPISIMO, and TXD1 will be connected to SPISTE, the chip select.

**Figure 28-11. FSITX as SPI Master, Transmit Only**



When the FSI is a SPI transmitter, the application has the ability to check for frame errors, line breaks, CRC errors, and ECC checks on data. These are all encoded by hardware in every FSI frame. The SPI receiver will require some software to act upon this information.

**Table 28-12. FSI as Master Transmitter, SPI as Slave Receiver**

Capability	Availability	Comment
Framing checks on the data frames	Yes	Can be implemented in software on the SPI receiver.

**Table 28-12. FSI as Master Transmitter, SPI as Slave Receiver (continued)**

Capability	Availability	Comment
Ability to detect line breaks	Yes	Can be implemented in software on the SPI receiver but will require additional software overhead such as a timer, or watchdog.
CRC check	Yes	Can be implemented in software on the SPI receiver. For devices which have VCU, this will be more efficient.
ECC on data	Yes	Can be implemented in software in the SPI receiver
Detection of abruptly terminated frames	No	
Double edge data rate	No	
Recovery from glitches on signal lines between frames	No	
Skew adjustment on signal lines	No	

**28.4.9.1.1 Initialization**

To configure the FSITX module to be a SPI master for transmit only, proceed through the standard FSITX initialization procedure. Before releasing the FSITX from reset, set TX\_OPER\_CTRL\_LO.SPI\_MODE to 1. This will enable the SPI clocking scheme and signaling structure.

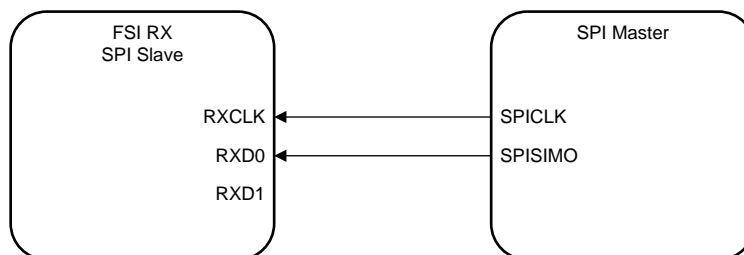
**28.4.9.1.2 Operation**

The operation of the FSITX module in SPI Compatibility mode is the same as if the module is in standard FSI mode. The application can utilize the frame timer, ping frames, external frame triggers, and so on. Refer to [Section 28.4.2](#) for more information on each of these features.

**28.4.9.1.2 FSIRX as SPI Slave, Receive Only**

The FSIRX can operate as an independent SPI slave module. In the usage, RXCLK will be connected to SPICLK, and RXD0 will be connected to SPISIMO. RXD1 is unused. There is no requirement for a chip select signal to be used when connected to the FSIRX. This is because the FSIRX will respond to any incoming clock edge. If there is any noise or unwanted clock transitions, a flush sequence will be required to resynchronize the FSIRX module with the Master.

**Figure 28-12. FSIRX as SPI Slave, Receive Only**



When the FSI is a SPI receiver communicating with a SPI transmitter, the application has the ability to detect frame errors, line breaks, CRC errors, ECC checks on data, as well as abruptly terminated frames. Note that the FSI can handle all of this in hardware, but the SPI transmitter must encode the information into the data to be transmitted.

**Table 28-13. SPI as master transmitter, FSI as slave receiver**

Capability	Availability	Comment
Framing checks on the data frames	Yes	Standard on FSI
Ability to detect line breaks	Yes	Can be implemented in software in the SPI transmitter. But will have to use some timer or watchdog in the transmitting SPI device
CRC check	Yes	Can be implemented in software in the SPI transmitter.
ECC on data	Yes	Can be implemented in software in the SPI transmitter.
Detection of abruptly terminated frames	Yes	This is accomplished with the FSI setting up the frame watchdog counter.
Double edge data rate	No	
Recovery from glitches on signal lines between frames	Yes	Whenever glitches occur on either the clock or data lines in between transmissions, the initial flush pattern of a frame will discard the effects of these glitches and will cause the receiver to resynchronize when the real "start-of-frame" pattern is seen. So, the ability to reject glitches in between frames is hence very high.
Skew adjustment on signal lines	Yes	The FSI receiver has the ability to add delays to the incoming signal lines.

#### 28.4.9.1.2.1 Initialization

To configure the FSIRX module to be a SPI slave for receiving only, proceed through the standard FSIRX initialization procedure. Before releasing the FSIRX from reset, set `RX_OPER_CTRL.SPI_MODE` to 1. This will enable the SPI clocking scheme and signaling structure.

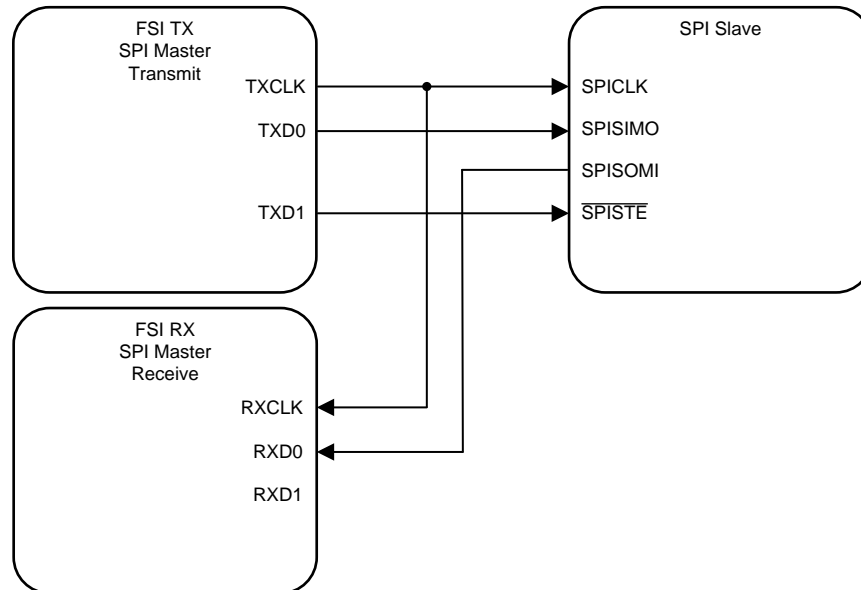
#### 28.4.9.1.2.2 Operation

The operation of the FSIRX module in SPI compatibility mode is the same as if the module is in standard FSI mode. The application can utilize the Frame and Ping Watchdogs, CRC and ECC checks, and so on. Refer to [Section 28.4.3](#) for more information on each of these features.

#### 28.4.9.1.3 FSITX and FSIRX Emulating a Full Duplex SPI Master

In this configuration, the FSITX is the clock master. The FSITX module will drive TXCLK (SPICLK), TXD0 (SPISIMO), and TXD1 (SPISTE/chip select), to the SPI slave. The SPISOMI signal will be connected back to the RXD0 signal. RXCLK can be fed either internally using the internal SPI pairing feature or wired externally, depending on the application requirements. Since the FSITX and RX modules are independent, the FSIRX could also be thought of as an additional SPI slave. Some software logic will be required in order for the FSI to emulate a SPI master fully.

**Figure 28-13. FSITX and FSIRX as SPI Master, Full Duplex**



### 28.4.9.1.3.1 Initialization

To configure both FSITX and RX modules for full duplex SPI master operation, follow the initialization instructions for each module described in the preceding sections. Both FSITX and RX modules must set their respective SPI\_MODE bits. This will enable the SPI clocking scheme and signaling structures.

If internal clock loopback is desired, the FSIRX module must also set RX\_MASTER\_CTRL.SPI\_PAIRING to 1. This will internally connect TXCLK to RXCLK. If using internal clock loopback, the GPIO used for RXCLK may be reallocated to other application requirements.

If the application requires an external clock loopback, ensure that TXCLK is connected to RXCLK. This may be required if the SPI Slave is across an isolation barrier and there is latency between TXCLK being launched and SPISOMI data being received on RXD0.

### 28.4.9.1.3.2 Operation

In this mode of operation, some higher level software must be written to emulate a full SPI master module. There is no path for the transmit module to determine what the receive module received. Both the TX and RX modules are still able to utilize the various other features available such as the ping frame timer, ping frame and frame watchdogs, CRC and ECC error checkers, and so on. The procedure for configuring these features is described elsewhere in this document.

## 28.5 Programmer's Model

This section describes various operational sequences and features for the FSI.

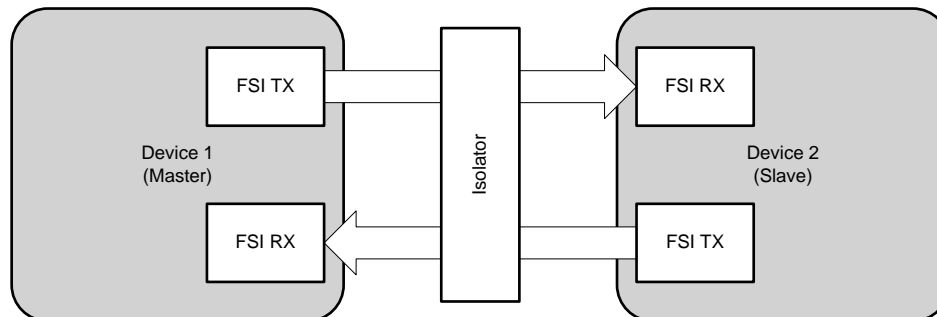
### 28.5.1 Establishing the Communication Link

Once the transmitter and receiver modules have been configured, some synchronization must occur before the modules may exchange data. Since the receiver will accept data on any clock transition, the receiver core logic must be flushed to properly interpret the start of a new, valid frame. This is especially true when the FSI modules reside on separate devices and are possibly isolated.

The following example walks through a suggested approach for establishing a clean communication link on two separate devices that may power up in an arbitrary order. Note that this is only a sample synchronization. Depending on application requirements, a different approach can be followed. The single, most important aspect of synchronization is to ensure that the receiver is properly flushed and ready to receive a complete frame without error. How to achieve this is up to the application.

Figure 28-14 shows the connection of the devices in this example. While there is no true concept of a master or a slave node in the FSI protocol, the example uses this nomenclature as a simple way to describe the data flow.

**Figure 28-14. Point to Point Connection**



Device 1 is the master node; it will be the driver of the initialization sequence. Device 2 is the slave node; it will respond to the master's commands. In this example, as well as in a real world use-case, neither the master nor the slave may know precisely when the other is ready to receive communication.

Sample sequences for both the master device and slave device are provided below:

#### 28.5.1.1 Establishing the Communication Link From the Master

The following sequence is an example of how the master node can establish the communication link with the slave without external signals outside of the standard communication link.

1. Assert the core reset to both the FSITX and FSIRX modules, and then deassert the resets.
2. Configure the transmitter and receiver for desired operation.
3. Set up the receiver interrupts to detect an incoming transmission.
4. Begin the ping loop:
  - Send the flush sequence.
  - Send a ping frame with the frame tag 0000.
  - Wait for some time. (determined by application)
  - If the FSIRX has received a valid ping frame, continue; else iterate the loop again.
  - If the received ping frame tag was 0001, continue; else iterate the loop again.
5. Send a ping frame with the frame tag 0001.

At this point both the master transmit and receive channels have successfully received a frame from their slave counterparts. The link has been established and standard application communication may begin.

#### 28.5.1.2 Establishing the Communication Link from the Slave

The following sequence is an example of how the slave node can establish the communication link with the master without external signals outside of the standard communication link.

1. Apply the core reset to both the FSITX and FSIRX modules, and then release the reset.
2. Configure the transmitter and receiver for desired operation.
3. Set up the receiver interrupts to detect an incoming transmission.
4. Wait for a receiver interrupt.
5. If the FSIRX has received a valid ping frame, continue; else return to step 4.
6. If the received frame tag was 0000, continue; else discard the transmission and return to step 4.
7. Send the flush sequence.
8. Send a ping frame with the frame tag 0001.
9. Wait for a receiver interrupt.
10. If the FSIRX has received a valid ping frame, continue; else return to step 4.



11. If the received ping frame tag was 0001, continue; else if the received frame tag was 0000, return to step 9. This can happen if a second ping frame was already in transit before receiving the slave's response in step 8.

At this point, both the transmit and receive modules have successfully received ping frames from their master counterparts. The link has been established and regular communication may now proceed. The application may configure periodic ping frames from the transmitter, initialize the receiver's ping and frame watchdogs and begin the communication required by the application.

### 28.5.2 Register Protection

Both the FSITX and FSIRX modules contain control registers that have embedded write protection. This is accomplished through EALLOW, register keys, and a master register lock. These protections ensure that no spurious writes or unintentional modifications to these registers are accepted. Refer to the [Section 28.6](#) for the register and bit descriptions to find the list of registers with write protections available.

#### EALLOW Protection

EALLOW is a device-level register protection; refer to [Section 3.1](#) for more information on EALLOW. For those registers with EALLOW protection, the EALLOW bit should be set before modifying the register. The application should then clear the EALLOW bit to re-enable the write protection when access to EALLOW-protected registers are complete.

#### Register Key Protection

In addition to EALLOW, some bits in the FSI registers are protected by a key. In order to write to these bits, the key must be written at the same time. For example, to put the transmitter core into reset, TX\_MASTER\_CTRL.CORE\_RST must be set. To do this, write 0xA501 to TX\_MASTER\_CTRL, where 0xA500 is the KEY value, and 0x0001 is the CORE\_RESET bit. Refer to the register descriptions for more information on which registers have write keys added.

#### Control Register Lock Protection

There also exists a master lock to prevent any modifications to the control registers. There is an independent lock for each FSI module. For the list of registers that are protected by this control register lock, refer to the register descriptions section. The control register lock will prevent any writes to the control registers until the lock is released. To set the control register lock, write 0xA501 to RX\_LOCK\_CTRL and TX\_LOCK\_CTRL for the receiver and transmitter respectively.

The control register lock cannot be disabled by the application until a SYSRSn has been asserted. This can occur at the device level, or by writing to the appropriate peripheral soft reset register (DEV\_CFG\_REGS.SOFTPRESx) for the FSI module. Refer to [Section 28.4.2.7](#) for more information on SYSRSn.

### 28.5.3 Emulation Mode

There is no specific emulation mode or configuration supported. The FSI cores will always be in free running mode. CPU halts will not have any effect on the operation of the FSI. However, reads of registers and data buffers by the debugger will not affect any flags, or status of the data buffers.

If the user wishes to stop the operation of either FSI module when the debugger halts, the following steps are required:

1. Set the debugger to real-time emulation mode.
2. Mark the FSI interrupt group as a time-critical interrupt. That is, enable the corresponding bit in the DBGIER register.
3. The ISR can check the DSTAT register and to determine if the ISR was called when the debugger was halted.
4. FSI operations can be disabled and the ISR can branch to a debug-specific halt location.

## 28.6 FSI Registers

This section describes the Fast Serial Interface Registers. The FSI module contains two distinct sets of registers. One for the FSI receiver, and another for the FSI transmitter.

### 28.6.1 FSI Base Addresses

**Table 28-14. FSI Base Address Table**

Device Registers	Register Name	Start Address	End Address
FsiTxaRegs	FSI_TX_REGS	0x0000_6600	0x0000_667F
FsiRxaRegs	FSI_RX_REGS	0x0000_6680	0x0000_66FF

## 28.6.2 FSI\_RX\_REGS Registers

Table 28-15 lists the FSI\_RX\_REGS registers. All register offset addresses not listed in Table 28-15 should be considered as reserved locations and the register contents should not be modified.

**Table 28-15. FSI\_RX\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	RX_MASTER_CTRL	Receive master control register	EALLOW	<a href="#">Go</a>
4h	RX_OPER_CTRL	Receive operation control register	EALLOW and LOCK	<a href="#">Go</a>
6h	RX_FRAME_INFO	Receive frame control register		<a href="#">Go</a>
7h	RX_FRAME_TAG_UDATA	Receive frame tag and user data register		<a href="#">Go</a>
8h	RX_DMA_CTRL	Receive DMA event control register	EALLOW and LOCK	<a href="#">Go</a>
Ah	RX_EVT_STS	Receive event and error status flag register		<a href="#">Go</a>
Bh	RX_CRC_INFO	Receive CRC info of received and computed CRC		<a href="#">Go</a>
Ch	RX_EVT_CLR	Receive event and error clear register	EALLOW	<a href="#">Go</a>
Dh	RX_EVT_FRC	Receive event and error flag force register	EALLOW	<a href="#">Go</a>
Eh	RX_BUF_PTR_LOAD	Receive buffer pointer load register	EALLOW	<a href="#">Go</a>
Fh	RX_BUF_PTR_STS	Receive buffer pointer status register		<a href="#">Go</a>
10h	RX_FRAME_WD_CTRL	Receive frame watchdog control register	EALLOW and LOCK	<a href="#">Go</a>
12h	RX_FRAME_WD_REF	Receive frame watchdog counter reference	EALLOW and LOCK	<a href="#">Go</a>
14h	RX_FRAME_WD_CNT	Receive frame watchdog current count		<a href="#">Go</a>
16h	RX_PING_WD_CTRL	Receive ping watchdog control register	EALLOW and LOCK	<a href="#">Go</a>
17h	RX_PING_TAG	Receive ping tag register		<a href="#">Go</a>
18h	RX_PING_WD_REF	Receive ping watchdog counter reference	EALLOW and LOCK	<a href="#">Go</a>
1Ah	RX_PING_WD_CNT	Receive pingwatchdog current count		<a href="#">Go</a>
1Ch	RX_INT1_CTRL	Receive interrupt control register for RX_INT1	EALLOW and LOCK	<a href="#">Go</a>
1Dh	RX_INT2_CTRL	Receive interrupt control register for RX_INT2	EALLOW and LOCK	<a href="#">Go</a>
1Eh	RX_LOCK_CTRL	Receive lock control register		<a href="#">Go</a>
20h	RX_ECC_DATA	Receive ECC data register		<a href="#">Go</a>
22h	RX_ECC_VAL	Receive ECC value register		<a href="#">Go</a>
24h	RX_ECC_SEC_DATA	Receive ECC corrected data register		<a href="#">Go</a>
26h	RX_ECC_LOG	Receive ECC log and status register		<a href="#">Go</a>
30h	RX_DLYLINE_CTRL	Receive delay line control register	EALLOW and LOCK	<a href="#">Go</a>
38h	RX_VIS_1	Receive debug visibility register 1		<a href="#">Go</a>
40h + formula	RX_BUF_BASE_y	Base address for receive data buffer	EALLOW and LOCK	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 28-16 shows the codes that are used for access types in this section.

**Table 28-16. FSI\_RX\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read

**Table 28-16. FSI\_RX\_REGS Access Type Codes (continued)**

Access Type	Code	Description
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**28.6.2.1 RX\_MASTER\_CTRL Register (Offset = 0h) [reset = 0h]**

RX\_MASTER\_CTRL is shown in [Figure 28-15](#) and described in [Table 28-17](#).

Return to the [Summary Table](#).

Receive master control register

**Figure 28-15. RX\_MASTER\_CTRL Register**

15	14	13	12	11	10	9	8
KEY							
W-0h							
7	6	5	4	3	2	1	0
RESERVED					SPI_PAIRING	INT_LOOPBACK	CORE_RST
R-0h					R/W-0h	R/W-0h	R/W-0h

**Table 28-17. RX\_MASTER\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	KEY	W	0h	<p>Write Key.</p> <p>In order to write to this register, 0xA5 must be written to this field at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every change to this register.</p> <p>Reset type: SYSRSn</p>
7-3	RESERVED	R	0h	Reserved
2	SPI_PAIRING	R/W	0h	<p>Clock Pairing for SPI-like Behavior Enable bit</p> <p>This bit enables the internal clock pairing with the FSI TX module. This feature internally connects the TXCLK to RXCLK allowing the FSI TX module, acting as a SPI master, to clock data into the receiver and out of the transmitter like a standard SPI module. This configuration is valid when the Module is in SPI mode only (RX_OPER_CTRL.SPI_MODE = 1)</p> <p>0h (R/W) = SPI clock pairing is not enabled.</p> <p>1h (R/W) = SPI clock pairing is enabled. The RXCLK will be internally connected to the TXCLK of the corresponding FSI module.</p> <p>Note: The KEY field must contain 0xA5 for any write to this bit to take effect.</p> <p>Reset type: SYSRSn</p>
1	INT_LOOPBACK	R/W	0h	<p>Internal Loopback Enable bit</p> <p>This bit enables the internal loopback functionality of the FSI receiver. By enabling this bit, a mux will select the signals coming directly from the corresponding FSI transmitter module rather than from the pins.</p> <p>0h (R/W) = Internal loopback is disabled. The FSI RX module will receive signals coming from the pins.</p> <p>1h (R/W) = Internal loopback is enabled. The FSI RX module will receive signals from the directly from FSI TX module rather than the pins.</p> <p>Note: The KEY field must contain 0xA5 for any write to this bit to take effect.</p> <p>Reset type: SYSRSn</p>

**Table 28-17. RX\_MASTER\_CTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	CORE_RST	R/W	0h	<p>Receiver Master Core Reset bit</p> <p>This bit controls the receiver master core reset. In order to receive any frame, this bit must be cleared.</p> <p>0h (R/W) = Receiver core is not in reset and can receive frames. 1h (R/W) = Receiver core is held in reset.</p> <p>Note: The KEY field must contain 0xA5 for any write to this bit to take effect.</p> <p>Reset type: SYSRSn</p>

**28.6.2.2 RX\_OPER\_CTRL Register (Offset = 4h) [reset = 0h]**

RX\_OPER\_CTRL is shown in [Figure 28-16](#) and described in [Table 28-18](#).

Return to the [Summary Table](#).

Receive operation control register

**Figure 28-16. RX\_OPER\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED							PING_WD_RST_MODE
R-0h							R/W-0h
7	6	5	4	3	2	1	0
ECC_SEL	N_WORDS			SPI_MODE		DATA_WIDTH	
R/W-0h	R/W-0h			R/W-0h		R/W-0h	

**Table 28-18. RX\_OPER\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	PING_WD_RST_MODE	R/W	0h	<p>Ping Watchdog Timeout Mode Select bit</p> <p>This bit selects the mode by which the ping watchdog counter is reset. The watchdog counter can be reset and restarted only by ping frames or by any received frame.</p> <p>0h (R/W) = The ping watchdog counter will reset and restart only by ping frames.</p> <p>1h (R/W) = The ping watchdog counter will reset and restart by any received frame.</p> <p>Reset type: SYSRSn</p>
7	ECC_SEL	R/W	0h	<p>ECC Data Width Select bit</p> <p>This bit selects between whether the ECC computation is done on 16-bit or 32-bit words.</p> <p>0h (R/W) = 32-bit ECC is used.</p> <p>1h (R/W) = 16-bit ECC is used.</p> <p>Reset type: SYSRSn</p>
6-3	N_WORDS	R/W	0h	<p>Number of Words to Receive</p> <p>This field defines the number of words which will be received in a DATA_N_WORD frame. This is a user-defined field that must match the corresponding field in the transmitter. Set this bitfield to be one less than the number of words to be received. This value is only applicable when the frame type received is DATA_N_WORD.</p> <p>0h (R/W) = 1 data word frame (16-bit data).</p> <p>1h (R/W) = 2 data word frame (32-bit data).</p> <p>..</p> <p>Fh (R/W) = 16 data word frame (256-bit data).</p> <p>Reset type: SYSRSn</p>
2	SPI_MODE	R/W	0h	<p>SPI Mode Enable bit</p> <p>This bit enables and disables the SPI compatibility mode of the FSI RX. The received data must be formatted as an FSI frame in order for the data to properly be received. SPI compatibility mode will allow FSI RX to receive data that is sent using SPI signal format. Refer to the applicable section in the FSI TRM chapter for more information.</p> <p>0h (R/W) = FSI is in normal mode of operation.</p> <p>1h (R/W) = FSI is operating in SPI compatibility mode.</p> <p>Reset type: SYSRSn</p>

**Table 28-18. RX\_OPER\_CTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	DATA_WIDTH	R/W	0h	<p>Receive Data Width Select bit</p> <p>These bits decide the number of data lines used for receiving data.</p> <p>0h (R/W) = Data will be received on one data line, RXD0.</p> <p>1h (R/W) = Data will be received on two data lines, RXD0 and RXD1.</p> <p>2h, 3h (R/W) = Reserved</p> <p>Reset type: SYSRSn</p>



**28.6.2.3 RX\_FRAME\_INFO Register (Offset = 6h) [reset = 0h]**

RX\_FRAME\_INFO is shown in [Figure 28-17](#) and described in [Table 28-19](#).

Return to the [Summary Table](#).

Receive frame control register

**Figure 28-17. RX\_FRAME\_INFO Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				FRAME_TYPE			
R-0h				R-0h			

**Table 28-19. RX\_FRAME\_INFO Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3-0	FRAME_TYPE	R	0h	<p>Received Frame Type</p> <p>This field indicates the type of frame that was successfully received last.</p> <p>0000b (R/W) = A ping frame was received</p> <p>0100b (R/W) = A DATA_1_WORD frame was received (16-bit data).</p> <p>0101b (R/W) = A DATA_2_WORD frame was received (32-bit data).</p> <p>0110b (R/W) = A DATA_4_WORD frame was received (64-bit data).</p> <p>0111b (R/W) = A DATA_6_WORD frame was received (96-bit data).</p> <p>0011b (R/W) = A DATA_N_WORD frame was received. The N_WORD field will determine the number of words (1 to 16) to be sent. The number of words received must equal the value programmed in RX_OPER_CTRL.N_WORDS.</p> <p>1111b (R/W) = An error frame was received. This frame can be used during error conditions or any condition where the transmitter wants to signal the receiver for attention. However, the user software is at liberty to use this for any purpose.</p> <p>0001b, 0010b, and 1000b through 1110b are Reserved and should not be used.</p> <p>Reset type: SYSRSn</p>

**28.6.2.4 RX\_FRAME\_TAG\_UDATA Register (Offset = 7h) [reset = 0h]**

RX\_FRAME\_TAG\_UDATA is shown in [Figure 28-18](#) and described in [Table 28-20](#).

Return to the [Summary Table](#).

Receive frame tag and user data register

**Figure 28-18. RX\_FRAME\_TAG\_UDATA Register**

15	14	13	12	11	10	9	8
USER_DATA							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FRAME_TAG				RESERVED
R-0h			R-0h				

**Table 28-20. RX\_FRAME\_TAG\_UDATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	USER_DATA	R	0h	Received User Data This field contains the 8-bit user data field of the last successfully received frame. Reset type: SYSRSn
7-5	RESERVED	R	0h	Reserved
4-1	FRAME_TAG	R	0h	Received Frame Tag This field contains the 4-bit frame tag from the last successfully received frame. This is intentionally shifted into bits 4:1 so that the register can be used as a 32-bit address index based on the received tag. Reset type: SYSRSn
0	RESERVED	R	0h	Reserved

**28.6.2.5 RX\_DMA\_CTRL Register (Offset = 8h) [reset = 0h]**

RX\_DMA\_CTRL is shown in [Figure 28-19](#) and described in [Table 28-21](#).

Return to the [Summary Table](#).

Receive DMA event control register

**Figure 28-19. RX\_DMA\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							DMA_EVT_EN
R-0h							R/W-0h

**Table 28-21. RX\_DMA\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	DMA_EVT_EN	R/W	0h	<p>DMA Event Enable bit</p> <p>This bit will enable a DMA Event to be generated upon the completion of a frame reception.</p> <p>0h (R/W) = A DMA event will not be generated.</p> <p>1h (R/W) = A DMA event will be generated upon the reception of a frame.</p> <p>Note: The DMA event will only be generated for data frames.</p> <p>Reset type: SYSRSn</p>

### 28.6.2.6 RX\_EVT\_STS Register (Offset = Ah) [reset = 0h]

RX\_EVT\_STS is shown in [Figure 28-20](#) and described in [Table 28-22](#).

Return to the [Summary Table](#).

Receive event and error status flag register

**Figure 28-20. RX\_EVT\_STS Register**

15	14	13	12	11	10	9	8
RESERVED				DATA_FRAME	FRAME_OVER RUN	PING_FRAME	ERR_FRAME
R-0h				R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
BUF_UNDE RRUN	FRAME_D ONE	BUF_OVE RRUN	EOF_ERR	TYPE_ERR	CRC_ERR	FRAME_WD T O	PING_WD T O
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 28-22. RX\_EVT\_STS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	DATA_FRAME	R	0h	Data Frame Received Flag This bit indicates that a data frame has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register. 0h (R) = No data frame has been received. 1h (R) = A data frame has been received. To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn
10	FRAME_OVERRUN	R	0h	Frame Overrun Flag This bit indicates that a frame overrun condition has occurred. This bit will be set when a new frame has been received while the FRAME_DONE flag is still set. Software can also force this bit to get set by writing to the RX_EVT_FRC register. 0h (R) = Frame overrun has not occurred. 1h (R) = Frame overrun has occurred. To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn
9	PING_FRAME	R	0h	Ping Frame Received Flag This bit indicates that a ping frame has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register. 0h (R) = No ping frame has been received. 1h (R) = A ping frame has been received. To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn

**Table 28-22. RX\_EVT\_STS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	ERR_FRAME	R	0h	<p>Error Frame Received Flag</p> <p>This bit indicates that an error frame has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register.</p> <p>0h (R) = No error frame has been received. 1h (R) = An error frame has been received.</p> <p>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn</p>
7	BUF_UNDERRUN	R	0h	<p>Receive Buffer Underrun Flag</p> <p>This bit indicates that a buffer underrun condition has occurred in the receive buffer. This will happen when software reads the buffer which is empty and has no valid data. Software can also force this bit to get set by writing to the RX_EVT_FRC register.</p> <p>0h (R) = Receive Buffer Underrun has not occurred. 1h (R) = Receive Buffer Underrun has occurred.</p> <p>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn</p>
6	FRAME_DONE	R	0h	<p>Frame Done Flag</p> <p>This bit indicates that a frame has been successfully received without error. Software can also force this bit to get set by writing to the RX_EVT_FRC register.</p> <p>0h (R) = No frame has been successfully received. 1h (R) = A frame has been successfully received.</p> <p>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn</p>
5	BUF_OVERRUN	R	0h	<p>Receive Buffer Overrun Flag</p> <p>This bit indicates that a buffer overrun condition has occurred in the receive buffer. Software can also force this bit to get set by writing to the RX_EVT_FRC register.</p> <p>0h (R) = Receive buffer overrun has not occurred. 1h (R) = Receive buffer overrun has occurred.</p> <p>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn</p>
4	EOF_ERR	R	0h	<p>End-of-Frame Error Flag</p> <p>This bit indicates that an invalid end-of-frame bit pattern has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register.</p> <p>0h (R) = Invalid end-of-frame has not been received. 1h (R) = Invalid end-of-frame has been received</p> <p>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn</p>

**Table 28-22. RX\_EVT\_STS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	TYPE_ERR	R	0h	<p>Frame Type Error Flag</p> <p>This bit indicates that an invalid frame type has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register.</p> <p>0h (R) = Invalid frame type has not been received. 1h (R) = Invalid frame type has been received</p> <p>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn</p>
2	CRC_ERR	R	0h	<p>CRC Error Flag</p> <p>This bit indicates that a CRC error has occurred. A CRC error will be generated on a data frame where the received CRC and the computed CRC do not match. Software can also force this bit to get set by writing to the RX_EVT_FRC register.</p> <p>0h (R) = CRC error has not occurred. 1h (R) = CRC error has occurred.</p> <p>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn</p>
1	FRAME_WD_TO	R	0h	<p>Frame Watchdog Timeout Flag</p> <p>This bit indicates that the frame watchdog timer has timed out. Software can also force this bit to get set by writing to the RX_EVT_FRC register.</p> <p>0h (R) = Frame watchdog timeout has not occurred. 1h (R) = Frame watchdog timeout has occurred.</p> <p>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn</p>
0	PING_WD_TO	R	0h	<p>Ping Watchdog Timeout Flag</p> <p>This bit indicates that the ping watchdog timer has timed out. Software can also force this bit to get set by writing to the RX_EVT_FRC register.</p> <p>0h (R) = Ping watchdog timeout has not occurred. 1h (R) = Ping watchdog timeout has occurred.</p> <p>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register. Reset type: SYSRSn</p>

**28.6.2.7 RX\_CRC\_INFO Register (Offset = Bh) [reset = 0h]**

RX\_CRC\_INFO is shown in [Figure 28-21](#) and described in [Table 28-23](#).

Return to the [Summary Table](#).

Receive CRC info of received and computed CRC

**Figure 28-21. RX\_CRC\_INFO Register**

15	14	13	12	11	10	9	8
CALC_CRC							
R-0h							
7	6	5	4	3	2	1	0
RX_CRC							
R-0h							

**Table 28-23. RX\_CRC\_INFO Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	CALC_CRC	R	0h	<p>Hardware Calculated CRC Value</p> <p>This bitfield contains the CRC value that was calculated on the last received data. The contents of this bitfield are valid only when data frames are received.</p> <p>Note: The contents of this bitfield are invalid for ping and error frames.</p> <p>Reset type: SYSRSn</p>
7-0	RX_CRC	R	0h	<p>Received CRC Value</p> <p>This bitfield contains the CRC value that was last received a frame. The contents of this bitfield are valid only when data frames are received.</p> <p>Note: The contents of this bitfield are invalid for ping and error frames.</p> <p>Reset type: SYSRSn</p>

**28.6.2.8 RX\_EVT\_CLR Register (Offset = Ch) [reset = 0h]**

 RX\_EVT\_CLR is shown in [Figure 28-22](#) and described in [Table 28-24](#).

 Return to the [Summary Table](#).

Receive event and error clear register

**Figure 28-22. RX\_EVT\_CLR Register**

15	14	13	12	11	10	9	8
RESERVED				DATA_FRAME	FRAME_OVER RUN	PING_FRAME	ERR_FRAME
R-0h				W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
BUF_UNDERR UN	FRAME_DONE	BUF_OVERRU N	EOF_ERR	TYPE_ERR	CRC_ERR	FRAME_WD_T O	PING_WD_TO
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

**Table 28-24. RX\_EVT\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	DATA_FRAME	W	0h	Data Frame Received Flag Clear bit This bit clears the corresponding bit in the RX_EVT_STS register. 0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0. Reset type: SYSRSn
10	FRAME_OVERRUN	W	0h	Frame Overrun Flag Clear bit This bit clears the corresponding bit in the RX_EVT_STS register. 0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0. Reset type: SYSRSn
9	PING_FRAME	W	0h	Ping Frame Received Flag Clear bit This bit clears the corresponding bit in the RX_EVT_STS register. 0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0. Reset type: SYSRSn
8	ERR_FRAME	W	0h	Error Frame Received Flag Clear bit This bit clears the corresponding bit in the RX_EVT_STS register. 0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0. Reset type: SYSRSn
7	BUF_UNDERRUN	W	0h	Receive Buffer Underrun Flag Clear bit This bit clears the corresponding bit in the RX_EVT_STS register. 0h (R/W) = Writing a 0 to this bit will have no effect. 1h (R/W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0. Reset type: SYSRSn



**Table 28-24. RX\_EVT\_CLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	FRAME_DONE	W	0h	<p>Frame Done Flag Clear bit</p> <p>This bit clears the corresponding bit in the RX_EVT_STS register.</p> <p>0h (W) = Writing a 0 to this bit will have no effect.</p> <p>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.</p> <p>Reset type: SYSRSn</p>
5	BUF_OVERRUN	W	0h	<p>Receive Buffer Overrun Flag Clear bit</p> <p>This bit clears the corresponding bit in the RX_EVT_STS register.</p> <p>0h (W) = Writing a 0 to this bit will have no effect.</p> <p>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.</p> <p>Reset type: SYSRSn</p>
4	EOF_ERR	W	0h	<p>End-of-Frame Error Flag Clear bit</p> <p>This bit clears the corresponding bit in the RX_EVT_STS register.</p> <p>0h (W) = Writing a 0 to this bit will have no effect.</p> <p>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.</p> <p>Reset type: SYSRSn</p>
3	TYPE_ERR	W	0h	<p>Frame Type Error Flag Clear bit</p> <p>This bit clears the corresponding bit in the RX_EVT_STS register.</p> <p>0h (W) = Writing a 0 to this bit will have no effect.</p> <p>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.</p> <p>Reset type: SYSRSn</p>
2	CRC_ERR	W	0h	<p>CRC Error Flag Clear bit</p> <p>This bit clears the corresponding bit in the RX_EVT_STS register.</p> <p>0h (W) = Writing a 0 to this bit will have no effect.</p> <p>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.</p> <p>Reset type: SYSRSn</p>
1	FRAME_WD_TO	W	0h	<p>Frame Watchdog Timeout Flag Clear bit</p> <p>This bit clears the corresponding bit in the RX_EVT_STS register.</p> <p>0h (W) = Writing a 0 to this bit will have no effect.</p> <p>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.</p> <p>Reset type: SYSRSn</p>
0	PING_WD_TO	W	0h	<p>Ping Watchdog Timeout Flag Clear bit</p> <p>This bit clears the corresponding bit in the RX_EVT_STS register.</p> <p>0h (W) = Writing a 0 to this bit will have no effect.</p> <p>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.</p> <p>Reset type: SYSRSn</p>

**28.6.2.9 RX\_EVT\_FRC Register (Offset = Dh) [reset = 0h]**

RX\_EVT\_FRC is shown in [Figure 28-23](#) and described in [Table 28-25](#).

Return to the [Summary Table](#).

Receive event and error flag force register

**Figure 28-23. RX\_EVT\_FRC Register**

15	14	13	12	11	10	9	8
RESERVED				DATA_FRAME	FRAME_OVER RUN	PING_FRAME	ERR_FRAME
R-0h				W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
BUF_UNDE RRUN	FRAME_D ONE	BUF_OVE RRUN	EOF_ERR	TYPE_ERR	CRC_ERR	FRAME_W D_T O	PING_W D_T O
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

**Table 28-25. RX\_EVT\_FRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	DATA_FRAME	W	0h	Data Frame Received Flag Force bit This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR. 0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn
10	FRAME_OVERRUN	W	0h	Frame Overrun Flag Force bit This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR. 0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn
9	PING_FRAME	W	0h	Ping Frame Received Flag Force bit This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR. 0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn
8	ERR_FRAME	W	0h	Error Frame Received Flag Force bit This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR. 0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn

**Table 28-25. RX\_EVT\_FRC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	BUF_UNDERRUN	W	0h	<p>Receive Buffer Underrun Flag Force bit</p> <p>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn</p>
6	FRAME_DONE	W	0h	<p>Frame Done Flag Force bit</p> <p>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn</p>
5	BUF_OVERRUN	W	0h	<p>Receive Buffer Overrun Flag Force bit</p> <p>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn</p>
4	EOF_ERR	W	0h	<p>End-of-Frame Error Flag Force bit</p> <p>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn</p>
3	TYPE_ERR	W	0h	<p>Frame Type Error Flag Force bit</p> <p>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn</p>
2	CRC_ERR	W	0h	<p>CRC Error Flag Force bit</p> <p>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn</p>
1	FRAME_WD_TO	W	0h	<p>Frame Watchdog Timeout Flag Force bit</p> <p>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding bit in the RX_EVT_STS Register. Reset type: SYSRSn</p>

**Table 28-25. RX\_EVT\_FRC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	PING_WD_TO	W	0h	<p>Ping Watchdog Timeout Flag Force bit</p> <p>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect.</p> <p>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.</p> <p>Reset type: SYSRSn</p>

**28.6.2.10 RX\_BUF\_PTR\_LOAD Register (Offset = Eh) [reset = 0h]**

RX\_BUF\_PTR\_LOAD is shown in [Figure 28-24](#) and described in [Table 28-26](#).

Return to the [Summary Table](#).

Receive buffer pointer load register

**Figure 28-24. RX\_BUF\_PTR\_LOAD Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				BUF_PTR_LOAD			
R-0h				R/W-0h			

**Table 28-26. RX\_BUF\_PTR\_LOAD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3-0	BUF_PTR_LOAD	R/W	0h	<p>Buffer Pointer Load.</p> <p>This is the value to be loaded into the receive word pointer when written. This is to allow software to force the receiver to start storing the received data starting at a specific location in the buffer.</p> <p>NOTE: The value of the CURR_BUF_PTR in the RX_BUF_PTR_STS will not get reflected immediately. This will take effect only when there is a valid receive operation with incoming clocks after (3 RXCLK + 3 SYCLK) cycles.</p> <p>Reset type: SYSRSn</p>

**28.6.2.11 RX\_BUF\_PTR\_STS Register (Offset = Fh) [reset = 0h]**

RX\_BUF\_PTR\_STS is shown in [Figure 28-25](#) and described in [Table 28-27](#).

Return to the [Summary Table](#).

Receive buffer pointer status register

**Figure 28-25. RX\_BUF\_PTR\_STS Register**

15	14	13	12	11	10	9	8
RESERVED			CURR_WORD_CNT				
R-0h			R-0h				
7	6	5	4	3	2	1	0
RESERVED				CURR_BUF_PTR			
R-0h				R-0h			

**Table 28-27. RX\_BUF\_PTR\_STS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-8	CURR_WORD_CNT	R	0h	Words Available in the Receive Buffer This bitfield indicates the number of valid data words present in the receive buffer that have not been read by the application software. This bitfield is only valid when there is no active transfer. Note: This value will not be valid if there has been a buffer overrun or underrun condition. Reset type: SYSRSn
7-4	RESERVED	R	0h	Reserved
3-0	CURR_BUF_PTR	R	0h	Current Buffer Pointer Index This bitfield will show the current index of the buffer pointer. This value is only valid when there is no active transmission. Reset type: SYSRSn

**28.6.2.12 RX\_FRAME\_WD\_CTRL Register (Offset = 10h) [reset = 0h]**

 RX\_FRAME\_WD\_CTRL is shown in [Figure 28-26](#) and described in [Table 28-28](#).

 Return to the [Summary Table](#).

Receive frame watchdog control register

**Figure 28-26. RX\_FRAME\_WD\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						FRAME_WD_EN	FRAME_WD_CNT_RST
R-0h						R/W-0h	R/W-0h

**Table 28-28. RX\_FRAME\_WD\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1	FRAME_WD_EN	R/W	0h	Frame Watchdog Counter Enable bit This bit will enable or disable the frame watchdog counter. The counter (RX_FRAME_WD_CNT) will begin counting from 0 when a valid start-of-frame pattern is received. When the reference value (RX_FRAME_WD_REF) is reached, it will generate a frame watchdog timeout event (RX_EVT_STS.FRAME_WD_TO) and the counter value will reset to 0 and continue counting on the next valid start-of-frame.  0h (R/W) = The frame watchdog counter is disabled and not running. 1h (R/W) = The frame watchdog counter logic is enabled and running. Reset type: SYSRSn
0	FRAME_WD_CNT_RST	R/W	0h	Frame Watchdog Counter Reset bit This bit will reset the frame watchdog counter to 0. This bit will always be read as 0.  0h (R/W) = Writing a 0 to this bit has no effect. 1h (W) = The frame watchdog counter will be reset to 0. Reset type: SYSRSn

**28.6.2.13 RX\_FRAME\_WD\_REF Register (Offset = 12h) [reset = 0h]**

RX\_FRAME\_WD\_REF is shown in [Figure 28-27](#) and described in [Table 28-29](#).

Return to the [Summary Table](#).

Receive frame watchdog counter reference

**Figure 28-27. RX\_FRAME\_WD\_REF 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
FRAME_WD_REF																															
R/W-0h																															

**Table 28-29. RX\_FRAME\_WD\_REF Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	FRAME_WD_REF	R/W	0h	<p>Frame Watchdog Counter Reference Value</p> <p>This is the 32-bit reference value for the frame watchdog timeout counter. The counter will count up starting from 0 at a valid start-of-frame pattern and continue counting until this value is reached.</p> <p>Reset type: SYSRSn</p>



**28.6.2.14 RX\_FRAME\_WD\_CNT Register (Offset = 14h) [reset = 0h]**

RX\_FRAME\_WD\_CNT is shown in [Figure 28-28](#) and described in [Table 28-30](#).

Return to the [Summary Table](#).

Receive frame watchdog current count

**Figure 28-28. RX\_FRAME\_WD\_CNT 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
FRAME_WD_CNT																															
R-0h																															

**Table 28-30. RX\_FRAME\_WD\_CNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	FRAME_WD_CNT	R	0h	<p>Frame Watchdog Counter Value</p> <p>This is the 32-bit read-only register which shows the current value of the frame watchdog counter. This counter is reset to 0 in a variety of ways: A write to FRME_WD_CNT_RST, a match with FRAME_WD_REF, or the reception of a successful data frame.</p> <p>Reset type: SYSRSn</p>

**28.6.2.15 RX\_PING\_WD\_CTRL Register (Offset = 16h) [reset = 0h]**

RX\_PING\_WD\_CTRL is shown in [Figure 28-29](#) and described in [Table 28-31](#).

Return to the [Summary Table](#).

Receive ping watchdog control register

**Figure 28-29. RX\_PING\_WD\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						PING_WD_EN	PING_WD_RST
R-0h						R/W-0h	R/W-0h

**Table 28-31. RX\_PING\_WD\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1	PING_WD_EN	R/W	0h	Ping Watchdog Counter Enable bit This bit will enable or disable the ping watchdog counter. The counter (RX_PING_WD_CNT) will begin counting from 0 when it is enabled. When the reference value (RX_PING_WD_REF) is reached, it will generate a ping watchdog timeout event (RX_EVT_STS.PING_WD_TO) and the counter value will reset to 0, and resume counting  0h (R/W) = The ping watchdog counter is disabled and not running. 1h (R/W) = The ping watchdog counter logic is enabled and running. Reset type: SYSRSn
0	PING_WD_RST	R/W	0h	Ping Watchdog Counter Reset bit This bit will reset the ping watchdog counter to 0. This bit will always be read as 0.  0h (R/W) = Writing a 0 to this bit has no effect. 1h (W) = The ping watchdog counter will be reset to 0. Reset type: SYSRSn

**28.6.2.16 RX\_PING\_TAG Register (Offset = 17h) [reset = 0h]**

RX\_PING\_TAG is shown in [Figure 28-30](#) and described in [Table 28-32](#).

Return to the [Summary Table](#).

Receive ping tag register

**Figure 28-30. RX\_PING\_TAG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			PING_TAG				RESERVED
R-0h			R-0h				

**Table 28-32. RX\_PING\_TAG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4-1	PING_TAG	R	0h	Received Ping Frame Tag This field contains the 4-bit frame tag from the last successfully received ping frame. This is intentionally shifted into bits 4:1 so that the register can be used as a 32-bit address index based on the received tag. Reset type: SYSRSn
0	RESERVED	R	0h	Reserved

**28.6.2.17 RX\_PING\_WD\_REF Register (Offset = 18h) [reset = 0h]**

RX\_PING\_WD\_REF is shown in [Figure 28-31](#) and described in [Table 28-33](#).

Return to the [Summary Table](#).

Receive ping watchdog counter reference

**Figure 28-31. RX\_PING\_WD\_REF 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
PING_WD_REF																															
R/W-0h																															

**Table 28-33. RX\_PING\_WD\_REF Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	PING_WD_REF	R/W	0h	Ping Watchdog Counter Reference Value This is the 32-bit reference value for the ping watchdog timeout counter. The counter will count up starting from 0 and continue counting until this value is reached. Reset type: SYSRSn

**28.6.2.18 RX\_PING\_WD\_CNT Register (Offset = 1Ah) [reset = 0h]**

RX\_PING\_WD\_CNT is shown in [Figure 28-32](#) and described in [Table 28-34](#).

Return to the [Summary Table](#).

Receive pingwatchdog current count

**Figure 28-32. RX\_PING\_WD\_CNT 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
PING_WD_CNT																															
R-0h																															

**Table 28-34. RX\_PING\_WD\_CNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	PING_WD_CNT	R	0h	Ping Watchdog Counter Value This is the 32-bit read-only register which shows the current value of the ping watchdog counter. This counter is reset to 0 in a variety of ways: A write to PING_WD_RST, a match with PING_WD_REF, or the reception of a ping frame. Reset type: SYSRSn

**28.6.2.19 RX\_INT1\_CTRL Register (Offset = 1Ch) [reset = 0h]**

 RX\_INT1\_CTRL is shown in [Figure 28-33](#) and described in [Table 28-35](#).

 Return to the [Summary Table](#).

Receive interrupt control register for RX\_INT1

**Figure 28-33. RX\_INT1\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED				INT1_EN_DATA_FRAME	INT1_EN_FRAME_OVERRUN	INT1_EN_PING_FRAME	INT1_EN_ERR_FRAME
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
INT1_EN_UNDERRUN	INT1_EN_FRAME_DONE	INT1_EN_OVERRUN	INT1_EN_EOF_ERR	INT1_EN_TYPE_ERR	INT1_EN_CRC_ERR	INT1_EN_FRAME_WD_TO	INT1_EN_PING_WD_TO
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 28-35. RX\_INT1\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	INT1_EN_DATA_FRAME	R/W	0h	Enable Data Frame Received Interrupt to INT1 bit This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.  0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = A data frame received event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register Reset type: SYSRSn
10	INT1_EN_FRAME_OVERRUN	R/W	0h	Enable Frame Overrun Interrupt to INT1 bit This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.  0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = A frame overrun event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register Reset type: SYSRSn
9	INT1_EN_PING_FRAME	R/W	0h	Enable Ping Frame Received Interrupt to INT1 bit This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.  0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = A ping frame received event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register Reset type: SYSRSn
8	INT1_EN_ERR_FRAME	R/W	0h	Enable ERROR Frame Received Interrupt to INT1 bit This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.  0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = A error frame received event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register Reset type: SYSRSn

**Table 28-35. RX\_INT1\_CTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	INT1_EN_UNDERRUN	R/W	0h	<p>Enable Buffer Underrun Interrupt to INT1 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = A buffer underrun event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
6	INT1_EN_FRAME_DONE	R/W	0h	<p>Enable Frame Done Interrupt to INT1 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = A frame done event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
5	INT1_EN_OVERRUN	R/W	0h	<p>Enable Receive Buffer Overrun Interrupt to INT1 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = A receive buffer overrun event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
4	INT1_EN_EOF_ERR	R/W	0h	<p>Enable End-of-Frame Error Interrupt to INT1 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = An end-of-frame error event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
3	INT1_EN_TYPE_ERR	R/W	0h	<p>Enable Frame Type Error Interrupt to INT1 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = A frame type error event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
2	INT1_EN_CRC_ERR	R/W	0h	<p>Enable CRC Error Interrupt to INT1 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT1. 1h (R/W) = A CRC error will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>

**Table 28-35. RX\_INT1\_CTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INT1_EN_FRAME_WD_TO	R/W	0h	<p>Enable Frame Watchdog Timeout Interrupt to INT1 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT1.</p> <p>1h (R/W) = A frame watchdog timeout event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
0	INT1_EN_PING_WD_TO	R/W	0h	<p>Enable Ping Watchdog Timeout Interrupt to INT1 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT1.</p> <p>1h (R/W) = A ping watchdog timeout event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>



### 28.6.2.20 RX\_INT2\_CTRL Register (Offset = 1Dh) [reset = 0h]

RX\_INT2\_CTRL is shown in [Figure 28-34](#) and described in [Table 28-36](#).

Return to the [Summary Table](#).

Receive interrupt control register for RX\_INT2

**Figure 28-34. RX\_INT2\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED				INT2_EN_DATA_FRAME	INT2_EN_FRAME_OVERRUN	INT2_EN_PING_FRAME	INT2_EN_ERR_FRAME
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
INT2_EN_UNDERRUN	INT2_EN_FRAME_DONE	INT2_EN_OVERRUN	INT2_EN_EOF_ERR	INT2_EN_TYPE_ERR	INT2_EN_CRC_ERR	INT2_EN_FRAME_WD_TO	INT2_EN_PING_WD_TO
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 28-36. RX\_INT2\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	INT2_EN_DATA_FRAME	R/W	0h	Enable Data Frame Received Interrupt to INT2 bit This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.  0h (R/W) = This event will not trigger an interrupt on RX_INT2. 1h (R/W) = A data frame received event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register Reset type: SYSRSn
10	INT2_EN_FRAME_OVERRUN	R/W	0h	Enable Frame Overrun Interrupt to INT2 bit This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.  0h (R/W) = This event will not trigger an interrupt on RX_INT2. 1h (R/W) = A frame overrun event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register Reset type: SYSRSn
9	INT2_EN_PING_FRAME	R/W	0h	Enable Ping Frame Received Interrupt to INT2 bit This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.  0h (R/W) = This event will not trigger an interrupt on RX_INT2. 1h (R/W) = A ping frame received event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register Reset type: SYSRSn
8	INT2_EN_ERR_FRAME	R/W	0h	Enable Error Frame Received Interrupt to INT2 bit This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.  0h (R/W) = This event will not trigger an interrupt on RX_INT2. 1h (R/W) = A error frame received event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register Reset type: SYSRSn

**Table 28-36. RX\_INT2\_CTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	INT2_EN_UNDERRUN	R/W	0h	<p>Enable Buffer Underrun Interrupt to INT2 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT2.</p> <p>1h (R/W) = A buffer underrun event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
6	INT2_EN_FRAME_DONE	R/W	0h	<p>Enable Frame Done Interrupt to INT2 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT2.</p> <p>1h (R/W) = A frame done event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
5	INT2_EN_OVERRUN	R/W	0h	<p>Enable Buffer Overrun Interrupt to INT2 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT2.</p> <p>1h (R/W) = A buffer overrun event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
4	INT2_EN_EOF_ERR	R/W	0h	<p>Enable End-of-Frame Error Interrupt to INT2 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT2.</p> <p>1h (R/W) = An end-of-frame error event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
3	INT2_EN_TYPE_ERR	R/W	0h	<p>Enable Frame Type Error Interrupt to INT2 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT2.</p> <p>1h (R/W) = A frame type error event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
2	INT2_EN_CRC_ERR	R/W	0h	<p>Enable CRC Error Interrupt to INT2 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT2.</p> <p>1h (R/W) = A CRC error will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>

**Table 28-36. RX\_INT2\_CTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	INT2_EN_FRAME_WD_TO	R/W	0h	<p>Enable Frame Watchdog Timeout Interrupt to INT2 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT2.</p> <p>1h (R/W) = A frame watchdog timeout event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>
0	INT2_EN_PING_WD_TO	R/W	0h	<p>Enable Ping Watchdog Timeout Interrupt to INT2 bit</p> <p>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.</p> <p>0h (R/W) = This event will not trigger an interrupt on RX_INT2.</p> <p>1h (R/W) = A ping watchdog timeout event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register</p> <p>Reset type: SYSRSn</p>

**28.6.2.21 RX\_LOCK\_CTRL Register (Offset = 1Eh) [reset = 0h]**

RX\_LOCK\_CTRL is shown in [Figure 28-35](#) and described in [Table 28-37](#).

Return to the [Summary Table](#).

Receive lock control register

**Figure 28-35. RX\_LOCK\_CTRL Register**

15	14	13	12	11	10	9	8
KEY							
W-0h							
7	6	5	4	3	2	1	0
RESERVED							LOCK
R-0h							R/W-0h

**Table 28-37. RX\_LOCK\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	KEY	W	0h	Write Key. In order to write to this register, 0xA5 must be written to this field at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every change to this register. Reset type: SYSRSn
7-1	RESERVED	R	0h	Reserved
0	LOCK	R/W	0h	Control Register Lock Enable bit This bit locks the contents of all the receive control registers that support a lock protection. Once locked, further writes will not take effect until SYSRS unlocks the register. Once set, further writes even to this bit will be ignored. 0h (R/W) = Receive control registers can be modified and are not locked. 1h (R/W) = Receive control registers are locked and cannot be modified until this bit is cleared by SYSRS. Any further writes to this bit are ignored. Note: The KEY field must contain 0xA5 for any write to this bit to take effect. Reset type: SYSRSn

**28.6.2.22 RX\_ECC\_DATA Register (Offset = 20h) [reset = 0h]**

RX\_ECC\_DATA is shown in [Figure 28-36](#) and described in [Table 28-38](#).

Return to the [Summary Table](#).

Receive ECC data register

**Figure 28-36. RX\_ECC\_DATA 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
DATA_HIGH																DATA_LOW															
R/W-0h																R/W-0h															

**Table 28-38. RX\_ECC\_DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA_HIGH	R/W	0h	<p>Upper 16 bits of ECC Data</p> <p>Writing to this bitfield will cause the ECC logic to compute the ECC(SEC-DED) the entire 32-bit register and update TX_ECC_VAL register with the results. Software should write to these 16 bits of the register in a 32-bit write when needing to compute ECC for 32-bits for the full TX_ECC_DATA register.</p> <p>Reset type: SYSRSn</p>
15-0	DATA_LOW	R/W	0h	<p>Lower 16 bits of ECC Data</p> <p>Writing to this bitfield will cause the ECC logic to compute the ECC(SEC-DED) for these 16 bits and update the TX_ECC_VAL register with the results. Software should write to these register bits as a 16-bit write when needing to compute ECC for 16-bits.</p> <p>Reset type: SYSRSn</p>

**28.6.2.23 RX\_ECC\_VAL Register (Offset = 22h) [reset = 0h]**

RX\_ECC\_VAL is shown in [Figure 28-37](#) and described in [Table 28-39](#).

Return to the [Summary Table](#).

Receive ECC value register

**Figure 28-37. RX\_ECC\_VAL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		ECC_VAL					
R-0h		R/W-0h					

**Table 28-39. RX\_ECC\_VAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6-0	ECC_VAL	R/W	0h	ECC Value for SEC-DED check This field contains the ECC value to be used for SEC-DED either for 16-bit or 32-bit data in the RX_ECC_DATA register. Reset type: SYSRSn

**28.6.2.24 RX\_ECC\_SEC\_DATA Register (Offset = 24h) [reset = 0h]**

RX\_ECC\_SEC\_DATA is shown in [Figure 28-38](#) and described in [Table 28-40](#).

Return to the [Summary Table](#).

Receive ECC corrected data register

**Figure 28-38. RX\_ECC\_SEC\_DATA 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
SEC_DATA																															
R-0h																															

**Table 28-40. RX\_ECC\_SEC\_DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SEC_DATA	R	0h	<p>ECC Single Error Corrected Data</p> <p>The ECC corrected data will be available in this register. This value is valid only when there are no bit errors, or a single bit error was detected. Otherwise, the contents of this register are invalid and should not be used.</p> <p>Reset type: SYSRSn</p>

**28.6.2.25 RX\_ECC\_LOG Register (Offset = 26h) [reset = 3h]**

RX\_ECC\_LOG is shown in [Figure 28-39](#) and described in [Table 28-41](#).

Return to the [Summary Table](#).

Receive ECC log and status register

**Figure 28-39. RX\_ECC\_LOG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						MBE	SBE
R-0h						R-1h	R-1h

**Table 28-41. RX\_ECC\_LOG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1	MBE	R	1h	<p>Multiple Bit Errors Detected</p> <p>This bit indicates the occurrence of multiple bit errors. The data is corrupted and cannot be corrected. If this bit is set, the data present in RX_ECC_SEC_DATA is invalid and should not be used.</p> <p>0h (R) Multiple Bit Errors were not detected. Check the SBE bit for single bit errors.</p> <p>1h (R) Multiple Bit Errors were detected. The data is not able to be corrected. The value present in RX_ECC_SEC_DATA is invalid and should not be used.</p> <p>Reset type: SYSRSn</p>
0	SBE	R	1h	<p>Single Bit Error Detected</p> <p>This bit indicates the occurrence of a single bit error in the data. The data is autocorrected and placed into the RX_ECC_SEC_DATA register. This bit is valid only if MBE is 0.</p> <p>0h (R) No bit errors were detected. The value in RX_ECC_SEC_DATA is correct.</p> <p>1h (R) A single bit error was detected and corrected. The corrected data is present in RX_ECC_SEC_DATA.</p> <p>Reset type: SYSRSn</p>



**28.6.2.26 RX\_DLYLINE\_CTRL Register (Offset = 30h) [reset = 0h]**

RX\_DLYLINE\_CTRL is shown in [Figure 28-40](#) and described in [Table 28-42](#).

Return to the [Summary Table](#).

Receive delay line control register

**Figure 28-40. RX\_DLYLINE\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED	RXD1_DLY				RXD0_DLY		
R-0h	R/W-0h				R/W-0h		
7	6	5	4	3	2	1	0
RXD0_DLY			RXCLK_DLY				
R/W-0h			R/W-0h				

**Table 28-42. RX\_DLYLINE\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-10	RXD1_DLY	R/W	0h	Delay Line Tap Select for RXD1 This bitfield selects the number of delay elements inserted into the RXD1 path from the pin boundary to the receiver core.  0h (R/W) Zero delay elements are included in the RXD1 path. RXD1 is taken directly from the pin. 1h (R/W) One delay element is included in the RXD1 path. 2h (R/W) Two delay elements are included in the RXD1 path. ... 1Fh (R/W) 31 delay elements are included in the RXD1 path, the maximum. Reset type: SYSRSn
9-5	RXD0_DLY	R/W	0h	Delay Line Tap Select for RXD0 This bitfield selects the number of delay elements inserted into the RXD0 path from the pin boundary to the receiver core.  0h (R/W) Zero delay elements are included in the RXD0 path. RXD0 is taken directly from the pin. 1h (R/W) One delay element is included in the RXD0 path. 2h (R/W) Two delay elements are included in the RXD0 path. ... 1Fh (R/W) 31 delay elements are included in the RXD0 path, the maximum. Reset type: SYSRSn
4-0	RXCLK_DLY	R/W	0h	Delay Line Tap Select for RXCLK This bitfield selects the number of delay elements inserted into the RXCLK path from the pin boundary to the receiver core.  0h (R/W) Zero delay elements are included in the RXCLK path. RXCLK is taken directly from the pin. 1h (R/W) One delay element is included in the RXCLK path. 2h (R/W) Two delay elements are included in the RXCLK path. ... 1Fh (R/W) 31 delay elements are included in the RXCLK path, the maximum. Reset type: SYSRSn

**28.6.2.27 RX\_VIS\_1 Register (Offset = 38h) [reset = 0h]**

RX\_VIS\_1 is shown in [Figure 28-41](#) and described in [Table 28-43](#).

Return to the [Summary Table](#).

Receive debug visibility register 1

**Figure 28-41. RX\_VIS\_1 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				RX_CORE_ST S	RESERVED			
R-0h				R-0h	R-0h			

**Table 28-43. RX\_VIS\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	RX_CORE_STS	R	0h	Receiver Core Status bit  This bit indicates the status of the receiver core. If this bit is set, the receiver should undergo a reset and subsequent resynchronization with the transmitter. This bit will be always be set when the receiver has detected and end of frame error or a frame type error. This bit can also be set if the receiver becomes corrupted due to noise on the signal lines. If the receiver has experienced a ping watchdog or frame watchdog timeout, this bit should be read to determine if the cause was due to a corrupt transaction, thus putting the receiver core into an unrecoverable state.  Only a soft reset will reset the receiver core and thus reset this bit.  0h (R) The receiver core is operating normally. 1h (R) The receiver core has entered into an error state and should be reset. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

**28.6.2.28 RX\_BUF\_BASE\_y Register (Offset = 40h + formula) [reset = 0h]**

RX\_BUF\_BASE\_y is shown in [Figure 28-42](#) and described in [Table 28-44](#).

Return to the [Summary Table](#).

Base address for receive data buffer

Offset = 40h + (y \* 1h); where y = 0h to Fh

**Figure 28-42. RX\_BUF\_BASE\_y Register**

15	14	13	12	11	10	9	8
BASE_ADDRESS							
R-0h							
7	6	5	4	3	2	1	0
BASE_ADDRESS							
R-0h							

**Table 28-44. RX\_BUF\_BASE\_y Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	BASE_ADDRESS	R	0h	Receive Data Buffer Base Address This is the base address of the 16-word data buffer used by the receiver. Reset type: SYSRSn

### 28.6.3 FSI\_TX\_REGS Registers

Table 28-45 lists the FSI\_TX\_REGS registers. All register offset addresses not listed in Table 28-45 should be considered as reserved locations and the register contents should not be modified.

**Table 28-45. FSI\_TX\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	TX_MASTER_CTRL	Transmit master control register	EALLOW	<a href="#">Go</a>
2h	TX_CLK_CTRL	Transmit clock control register	EALLOW and LOCK	<a href="#">Go</a>
4h	TX_OPER_CTRL_LO	Transmit operation control register low	EALLOW and LOCK	<a href="#">Go</a>
5h	TX_OPER_CTRL_HI	Transmit operation control register high	EALLOW and LOCK	<a href="#">Go</a>
6h	TX_FRAME_CTRL	Transmit frame control register		<a href="#">Go</a>
7h	TX_FRAME_TAG_UDATA	Transmit frame tag and user data register		<a href="#">Go</a>
8h	TX_BUF_PTR_LOAD	Transmit buffer pointer control load register	EALLOW	<a href="#">Go</a>
9h	TX_BUF_PTR_STS	Transmit buffer pointer control status register		<a href="#">Go</a>
Ah	TX_PING_CTRL	Transmit ping control register	EALLOW and LOCK	<a href="#">Go</a>
Bh	TX_PING_TAG	Transmit ping tag register		<a href="#">Go</a>
Ch	TX_PING_TO_REF	Transmit ping timeout counter reference	YES	<a href="#">Go</a>
Eh	TX_PING_TO_CNT	Transmit ping timeout current count		<a href="#">Go</a>
10h	TX_INT_CTRL	Transmit interrupt event control register	EALLOW and LOCK	<a href="#">Go</a>
11h	TX_DMA_CTRL	Transmit DMA event control register	EALLOW and LOCK	<a href="#">Go</a>
12h	TX_LOCK_CTRL	Transmit lock control register	EALLOW and LOCK	<a href="#">Go</a>
14h	TX_EVT_STS	Transmit event and error status flag register		<a href="#">Go</a>
16h	TX_EVT_CLR	Transmit event and error clear register	EALLOW	<a href="#">Go</a>
17h	TX_EVT_FRC	Transmit event and error flag force register	EALLOW	<a href="#">Go</a>
18h	TX_USER_CRC	Transmit user-defined CRC register		<a href="#">Go</a>
20h	TX_ECC_DATA	Transmit ECC data register		<a href="#">Go</a>
22h	TX_ECC_VAL	Transmit ECC value register		<a href="#">Go</a>
40h + formula	TX_BUF_BASE_y	Base address for transmit buffer		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 28-46 shows the codes that are used for access types in this section.

**Table 28-46. FSI\_TX\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		

**Table 28-46. FSI\_TX\_REGS Access Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**28.6.3.1 TX\_MASTER\_CTRL Register (Offset = 0h) [reset = 0h]**

TX\_MASTER\_CTRL is shown in [Figure 28-43](#) and described in [Table 28-47](#).

Return to the [Summary Table](#).

Transmit master control register

**Figure 28-43. TX\_MASTER\_CTRL Register**

15	14	13	12	11	10	9	8
KEY							
W-0h							
7	6	5	4	3	2	1	0
RESERVED						FLUSH	CORE_RST
R-0h						R/W-0h	R/W-0h

**Table 28-47. TX\_MASTER\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	KEY	W	0h	Write Key In order to write to any bit in this register, 0xA5 must be written to this field at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every change to this register. Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	FLUSH	R/W	0h	Flush Operation Start bit This bit will cause the transmitter to initiate a flush pattern of a single toggle on the TXD0 and TXD1 followed by five full cycles of TXCLK. This bit should be written only when the CORE_RST bit is 0 and the clock to the Transmitter core is turned on. 0h (R/W) = Clear this bit. 1h (R/W) = Setting this bit will initiate flush sequence. To properly execute a flush sequence, Set FLUSH to 1, wait for five TXCLK cycles then clear FLUSH to 0. Note: The KEY field must contain 0xA5 for any write to this bit to take effect. The software must keep this bit set to 1 for at least five TXCLK cycles before setting it back to 0. Reset type: SYSRSn
0	CORE_RST	R/W	0h	Transmitter Master Core Reset bit This bit controls the transmitter master core reset. In order to send any frame, this bit must be cleared. 0h (R/W) = Transmitter core is not in reset and can transmit frames. 1h (R/W) = Transmitter core is held in reset. Note: The KEY field must contain 0xA5 for any write to this bit to take effect. Reset type: SYSRSn

### 28.6.3.2 TX\_CLK\_CTRL Register (Offset = 2h) [reset = 0h]

TX\_CLK\_CTRL is shown in [Figure 28-44](#) and described in [Table 28-48](#).

Return to the [Summary Table](#).

Transmit clock control register

**Figure 28-44. TX\_CLK\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED						PRESCALE_VAL	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
PRESCALE_VAL						CLK_EN	CLK_RST
R/W-0h						R/W-0h	R/W-0h

**Table 28-48. TX\_CLK\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-2	PRESCALE_VAL	R/W	0h	Clock Divider Prescale Value The input clock is divided by this 8-bit value and fed into the transmitter core. This divided clock is the rate at which TXCLK will operate.  0h (R/W) = Input clock /1 1h (R/W) = Input clock /1 2h (R/W) = Input clock /2 3h (R/W) = Input clock /3 4h (R/W) = Input clock /4 ... FFh (R/W) = Input clock /255  TXCLKIN = Input clock / PRESCALE_VAL In FSI mode: TXCLK = TXCLKIN / 2 In SPI mode: TXCLK = TXCLKIN Reset type: SYSRSn
1	CLK_EN	R/W	0h	Clock Divider Enable bit This bit will enable and disable the input clock divider and start the clock to the transmitter core.  0h (R/W) = The input clock divider is not enabled and the clock is not connected to the transmitter core. 1h (R/W) = The input clock to the transmitter core is being divided by the PRESCALE_VAL and enabled. Reset type: SYSRSn
0	CLK_RST	R/W	0h	Clock Divider Reset bit This bit will reset the clock counter in the clock divider.  0h (R/W) = The clock divider is set based on the value in PRESCALE_VAL. The input clock will be divided by PRESCALE_VAL if CLK_EN is set. 1h (R/W) = The clock divider will be reset to 0 and will stay reset until software writes a 0 to this bit. Reset type: SYSRSn

**28.6.3.3 TX\_OPER\_CTRL\_LO Register (Offset = 4h) [reset = 0h]**

TX\_OPER\_CTRL\_LO is shown in [Figure 28-45](#) and described in [Table 28-49](#).

Return to the [Summary Table](#).

Transmit operation control register low

**Figure 28-45. TX\_OPER\_CTRL\_LO Register**

15	14	13	12	11	10	9	8
RESERVED							SEL_PLLCLK
R-0h							R/W-0h
7	6	5	4	3	2	1	0
PING_TO_MODE	SW_CRC	START_MODE			SPI_MODE	DATA_WIDTH	
R/W-0h	R/W-0h	R/W-0h			R/W-0h	R/W-0h	

**Table 28-49. TX\_OPER\_CTRL\_LO Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	SEL_PLLCLK	R/W	0h	Input Clock Select bit This bit selects the input clock source for the transmitter core.  0h (R/W) = SYSCLK is the source of the transmitter clock into the clock prescaler. 1h (R/W) = PLLRAWCLK is the source of the transmitter core clock into the clock prescaler. Reset type: SYSRSn
7	PING_TO_MODE	R/W	0h	Ping Counter Reset Mode Select bit This bit selects when the ping counter will reset.  0h (R/W) = The ping counter will reset and restart only on hardware initiated ping frames, when ping counter has timed out. 1h (R/W) = The ping counter will reset and restart on any software initiated frame as well as a ping counter timeout Reset type: SYSRSn
6	SW_CRC	R/W	0h	CRC Source Select bit This bit selects the source of the CRC value that is transmitted.  0h (R/W) = The transmitted CRC value is computed by hardware. 1h (R/W) = The transmitted CRC value is sourced from the value programmed in the TX_USER_CRC register. Reset type: SYSRSn
5-3	START_MODE	R/W	0h	Transmission Start Mode Select bit These bits select the method by which a new frame transmission is started.  0h (R/W) = Only a software write to TX_FRAME_CTRL.START initiate a new transmission. 1h (R/W) = The configured external trigger will initiate a new transmission. A rising edge on the external trigger will initiate a start of transmission. The external trigger signal should be at least 3 SYSCLK cycles wide. 2h (R/W) = Either writing to TX_FRAME_CTRL.START or the TX_FRAME_TAG_UDATA register will initiate a new transmission.  All other combinations of bits are illegal and reserved for future use. Reset type: SYSRSn



**Table 28-49. TX\_OPER\_CTRL\_LO Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	SPI_MODE	R/W	0h	SPI Mode Select bit This bit enables and disables SPI compatibility mode. 0h (R/W) = FSI is in normal mode of operation. 1h (R/W) = FSI is operating in SPI compatibility mode. Reset type: SYSRSn
1-0	DATA_WIDTH	R/W	0h	Transmit Data Width Select bits These bits define the number of data lines used by the transmitter. 0h (R/W) = Data will be transmitted on one data line (TXD0) 1h (R/W) = Data will be transmitted on two data lines (TXD0 and TXD1). The format of the data is described in the preceding chapter. 2h, 3h (R/W) = Reserved Reset type: SYSRSn

**28.6.3.4 TX\_OPER\_CTRL\_HI Register (Offset = 5h) [reset = 0h]**

TX\_OPER\_CTRL\_HI is shown in [Figure 28-46](#) and described in [Table 28-50](#).

Return to the [Summary Table](#).

Transmit operation control register high

**Figure 28-46. TX\_OPER\_CTRL\_HI Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	ECC_SEL	FORCE_ERR	EXT_TRIG_SEL				
R-0h	R/W-0h	R/W-0h	R/W-0h				

**Table 28-50. TX\_OPER\_CTRL\_HI Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	ECC_SEL	R/W	0h	ECC Data Width Select bit This bit selects between 16-bit and 32-bit ECC computation. 0h (R/W) = 32-bit ECC is used. 1h (R/W) = 16-bit ECC is used. Reset type: SYSRSn
5	FORCE_ERR	R/W	0h	Error Frame Force bit This bit will force the the CRC value of the transmitted data frame to 0 whenever there is a buffer overrun or underrun condition. This can be used to force a corrupted CRC as the data is not guaranteed to be reliable. The receiver will treat the data as invalid and can handle this as needed. 0h (R/W) = The CRC will not be forced to 0. 1h (R/W) = The CRC will be forced to 0 in a buffer overrun or underrun condition. Reset type: SYSRSn
4-0	EXT_TRIG_SEL	R/W	0h	External Trigger Select bit These bits define which of the 32 external inputs will be used as the source for the external input trigger. 00h (R/W) = Trigger 1 is the source. 01h (R/W) = Trigger 2 is the source. 02h (R/W) = Trigger 3 is the source. ... 1Fh (R/W) = Trigger 32 is the source. Reset type: SYSRSn

### 28.6.3.5 TX\_FRAME\_CTRL Register (Offset = 6h) [reset = 0h]

TX\_FRAME\_CTRL is shown in [Figure 28-47](#) and described in [Table 28-51](#).

Return to the [Summary Table](#).

Transmit frame control register

**Figure 28-47. TX\_FRAME\_CTRL Register**

15	14	13	12	11	10	9	8
START		RESERVED					
R/W-0h		R-0h					
7	6	5	4	3	2	1	0
N_WORDS				FRAME_TYPE			
R/W-0h				R/W-0h			

**Table 28-51. TX\_FRAME\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	START	R/W	0h	Start Transmission bit This bit will cause the FSI to start transmitting the next frame.  0h (R/W) = Writing a 0 to this bit will have no effect. 1h (R/W) = Start the next transmission. This bit will be cleared by hardware. Reset type: SYSRSn
14-8	RESERVED	R	0h	Reserved
7-4	N_WORDS	R/W	0h	Number of Words to be Transmitted This field defines the number of words which will be transmitted in a DATA_N_WORD frame. This is a user-defined field that must match the corresponding field in the receiver. Set this bitfield to be one less than the number of words to be transmitted.  0h (R/W) = 1 data word frame (16-bit data). 1h (R/W) = 2 data word frame (32-bit data). .. Fh (R/W) = 16 data word frame (256-bit data). Reset type: SYSRSn

**Table 28-51. TX\_FRAME\_CTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	FRAME_TYPE	R/W	0h	<p>Transmit Frame Type</p> <p>This field determines the type of frame that will be transmitted next.</p> <p>0000b (R/W) = Ping Frame. This frame can be sent either by software or automatically by hardware.</p> <p>0100b (R/W) = DATA_1_WORD Frame. One word data frame (16-bit data).</p> <p>0101b (R/W) = DATA_2_WORD Frame. Two word data frame (32-bit data).</p> <p>0110b (R/W) = DATA_4_WORD Frame. Four word data frame (64-bit data).</p> <p>0111b (R/W) = DATA_6_WORD Frame. Six word data frame (96-bit data).</p> <p>0011b (R/W) = DATA_N_WORD Frame. The N_WORDS field will determine the number of words (1 to 16) to be sent. Both the transmitter and receiver must have the same value programmed.</p> <p>1111b (R/W) = Error Frame. This frame can be used during error conditions or any condition where the transmitter wants to notify the receiver of a high priority status. However, the user software is at liberty to use this for any purpose.</p> <p>0001b, 0010b, and 1000b through 1110b are Reserved and should not be used.</p> <p>Reset type: SYSRSn</p>

**28.6.3.6 TX\_FRAME\_TAG\_UDATA Register (Offset = 7h) [reset = 0h]**

TX\_FRAME\_TAG\_UDATA is shown in [Figure 28-48](#) and described in [Table 28-52](#).

Return to the [Summary Table](#).

Transmit frame tag and user data register

**Figure 28-48. TX\_FRAME\_TAG\_UDATA Register**

15	14	13	12	11	10	9	8
USER_DATA							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				FRAME_TAG			
R-0h				R/W-0h			

**Table 28-52. TX\_FRAME\_TAG\_UDATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	USER_DATA	R/W	0h	<p>User Data bits</p> <p>This is a user-defined value that will be loaded into the the user data phase of the frame. This 8-bit value can be used by the receiver for any application need. This value will not impact any hardware behavior.</p> <p>Reset type: SYSRSn</p>
7-4	RESERVED	R	0h	Reserved
3-0	FRAME_TAG	R/W	0h	<p>This will be used only for software initiated transmissions.</p> <p>Frame tag bits</p> <p>This is a user-defined value that will be loaded into the frame tag phase of the next transmission. The receiver may use the frame tag for any application need. This value will not impact any hardware behavior</p> <p>For external triggers do not use this register. Use the TX_PING_TAG register instead.</p> <p>Reset type: SYSRSn</p>

**28.6.3.7 TX\_BUF\_PTR\_LOAD Register (Offset = 8h) [reset = 0h]**

TX\_BUF\_PTR\_LOAD is shown in [Figure 28-49](#) and described in [Table 28-53](#).

Return to the [Summary Table](#).

Transmit buffer pointer control load register

**Figure 28-49. TX\_BUF\_PTR\_LOAD Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				BUF_PTR_LOAD			
R-0h				R/W-0h			

**Table 28-53. TX\_BUF\_PTR\_LOAD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3-0	BUF_PTR_LOAD	R/W	0h	Buffer Pointer Load bits These bits are used to force the transmit buffer pointer to a desired index within the transmit buffer. The next transmission will begin picking data from this index and increment appropriately. This value will be reflected in TX_BUF_PTR_STS only after a minimum 3 SYSCLK cycles + 3 TXCLK cycles.  This value should not be written while there is an active transmission as it may corrupt the ongoing frame or other undefined behavior. Reset type: SYSRSn

**28.6.3.8 TX\_BUF\_PTR\_STS Register (Offset = 9h) [reset = 0h]**

TX\_BUF\_PTR\_STS is shown in [Figure 28-50](#) and described in [Table 28-54](#).

Return to the [Summary Table](#).

Transmit buffer pointer control status register

**Figure 28-50. TX\_BUF\_PTR\_STS Register**

15	14	13	12	11	10	9	8
RESERVED			CURR_WORD_CNT				
R-0h			R-0h				
7	6	5	4	3	2	1	0
RESERVED				CURR_BUF_PTR			
R-0h				R-0h			

**Table 28-54. TX\_BUF\_PTR\_STS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-8	CURR_WORD_CNT	R	0h	Words Remaining in the transmit buffer This value indicates the number of words present in the data buffer which have not yet been transmitted. This value is only valid when there is no active transmission.  Note: This value will not be valid if there is a buffer overrun or underrun condition. Reset type: SYSRSn
7-4	RESERVED	R	0h	Reserved
3-0	CURR_BUF_PTR	R	0h	Current Buffer Pointer Index This bitfield will show the current index of the buffer pointer. This value is only valid when there is no active transmission. Reset type: SYSRSn

**28.6.3.9 TX\_PING\_CTRL Register (Offset = Ah) [reset = 0h]**

TX\_PING\_CTRL is shown in [Figure 28-51](#) and described in [Table 28-55](#).

Return to the [Summary Table](#).

Transmit ping control register

**Figure 28-51. TX\_PING\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
EXT_TRIG_SEL				EXT_TRIG_EN		TIMER_EN	CNT_RST
R/W-0h				R/W-0h		R/W-0h	R/W-0h

**Table 28-55. TX\_PING\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-3	EXT_TRIG_SEL	R/W	0h	External Trigger Select bits This bitfield will select one of the 32 external trigger inputs to as the source to generate a ping frame. A ping frame will only be generated if the EXT_TRIG_EN bit is set.  0h (R/W) = Trigger 1 will be used to generate a ping frame. 1h (R/W) = Trigger 2 will be used to generate a ping frame. .. 1Fh (R/W) = Trigger 32 will be used to generate a ping frame. Reset type: SYSRSn
2	EXT_TRIG_EN	R/W	0h	External Trigger Enable bit This bit will allow the external trigger logic to generate a ping frame.  0h (R/W) = External triggers will not be used to generate ping frames. 1h (R/W) = The selected external trigger (selected by EXT_TRIG_SEL bits) will be able to generate a ping frame. The ping timer will be ignored if this bit is set. Reset type: SYSRSn
1	TIMER_EN	R/W	0h	Ping Timer Enable bit This bit will enable the ping timer for generating periodic ping frames.  0h (R/W) = The ping timer is disabled and will not generate ping frames. 1h (R/W) = The ping timer is enabled and can be used to generate ping frames. Once the timer count reaches the value set by the TX_PING_TO_REF register, it will initiate a ping frame transmission.  Note: If the ping timer is used, EXT_TRIG_EN should not be set as it will override this function. Reset type: SYSRSn
0	CNT_RST	R/W	0h	Ping Counter Reset bit This bit will reset the the ping counter to 0. This bit will always be read as 0.  0h (R/W) = Writing a 0 to this bit has no effect. 1h (R/W) = The ping counter will be reset to 0. Reset type: SYSRSn



**28.6.3.10 TX\_PING\_TAG Register (Offset = Bh) [reset = 0h]**

TX\_PING\_TAG is shown in [Figure 28-52](#) and described in [Table 28-56](#).

Return to the [Summary Table](#).

Transmit ping tag register

**Figure 28-52. TX\_PING\_TAG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				TAG			
R-0h				R/W-0h			

**Table 28-56. TX\_PING\_TAG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3-0	TAG	R/W	0h	Ping Frame Tag This field contains a 4-bit tag which will be sent in any ping frame that is initiated by an external trigger or the ping timer. This field is user-defined and can be set based on the application requirement. If a ping frame is generated manually, the transmitted tag will be from TX_FRAME_TAG_UDATA.FRAME_TAG, not this value. Reset type: SYSRSn

**28.6.3.11 TX\_PING\_TO\_REF Register (Offset = Ch) [reset = 0h]**

TX\_PING\_TO\_REF is shown in [Figure 28-53](#) and described in [Table 28-57](#).

Return to the [Summary Table](#).

Transmit ping timeout counter reference

**Figure 28-53. TX\_PING\_TO\_REF 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
TO_REF																															
R/W-0h																															

**Table 28-57. TX\_PING\_TO\_REF Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	TO_REF	R/W	0h	Ping Timer Reference Value. This is the 32-bit reference value for the ping timer. The timer will increment the counter starting from 0. When the reference value is reached, it will generate a timeout event, triggering a ping frame transmission. The counter will then reset to 0 and continue counting. Reset type: SYSRSn

**28.6.3.12 TX\_PING\_TO\_CNT Register (Offset = Eh) [reset = 0h]**

TX\_PING\_TO\_CNT is shown in [Figure 28-54](#) and described in [Table 28-58](#).

Return to the [Summary Table](#).

Transmit ping timeout current count

**Figure 28-54. TX\_PING\_TO\_CNT 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
TO_CNT																															
R-0h																															

**Table 28-58. TX\_PING\_TO\_CNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	TO_CNT	R	0h	Ping Timer Counter Value This register contains the current value of the ping timer counter. After reset, this counter will increment until it reaches the reference value (TX_PING_TO_REF), at which point it generates a ping frame transmission. After this point, the counter will reset to 0 and continue counting. This is a free-running counter Reset type: SYSRSn

**28.6.3.13 TX\_INT\_CTRL Register (Offset = 10h) [reset = 0h]**

TX\_INT\_CTRL is shown in [Figure 28-55](#) and described in [Table 28-59](#).

Return to the [Summary Table](#).

Transmit interrupt event control register

**Figure 28-55. TX\_INT\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED				INT2_EN_PING_TO	INT2_EN_BUF_OVERRUN	INT2_EN_BUF_UNDERRUN	INT2_EN_FRAME_DONE
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED				INT1_EN_PING_TO	INT1_EN_BUF_OVERRUN	INT1_EN_BUF_UNDERRUN	INT1_EN_FRAME_DONE
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 28-59. TX\_INT\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	INT2_EN_PING_TO	R/W	0h	Enable PING Timer Interrupt to INT2 This bit allows the event to generate an interrupt on the INT2 line. 0h (R/W) = This event will not trigger an interrupt on TX_INT2. 1h (R/W) = The ping timer event will trigger an interrupt on TX_INT2. Reset type: SYSRSn
10	INT2_EN_BUF_OVERRUN	R/W	0h	Enable Buffer Overrun Interrupt to INT2 This bit allows the event to generate an interrupt on the INT2 line. 0h (R/W) = This event will not trigger an interrupt on TX_INT2. 1h (R/W) = A Buffer Overrun condition will trigger an interrupt on TX_INT2. Reset type: SYSRSn
9	INT2_EN_BUF_UNDERRUN	R/W	0h	Enable Buffer Underrun Interrupt to INT2 This bit allows the event to generate an interrupt on the INT2 line. 0h (R/W) = This event will not trigger an interrupt on TX_INT2. 1h (R/W) = A Buffer Underrun condition will trigger an interrupt on TX_INT2. Reset type: SYSRSn
8	INT2_EN_FRAME_DONE	R/W	0h	Enable Frame Done interrupt to INT2 This bit allows the event to generate an interrupt on the INT2 line. 0h (R/W) = This event will not trigger an interrupt on TX_INT2. 1h (R/W) = A Frame Done event will trigger an interrupt on TX_INT2. Reset type: SYSRSn
7-4	RESERVED	R	0h	Reserved
3	INT1_EN_PING_TO	R/W	0h	Enable Ping Timer Interrupt to INT1 This bit allows the event to generate an interrupt on the INT1 line. 0h (R/W) = This event will not trigger an interrupt on TX_INT1. 1h (R/W) = The ping timer event will trigger an interrupt on TX_INT1. Reset type: SYSRSn

**Table 28-59. TX\_INT\_CTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	INT1_EN_BUF_OVERRUN	R/W	0h	<p>Enable Buffer Overrun Interrupt to INT1</p> <p>This bit allows the event to generate an interrupt on the INT1 line.</p> <p>0h (R/W) = This event will not trigger an interrupt on TX_INT1.</p> <p>1h (R/W) = A Buffer Overrun condition will trigger an interrupt on TX_INT1.</p> <p>Reset type: SYSRSn</p>
1	INT1_EN_BUF_UNDERRUN	R/W	0h	<p>Enable Buffer Underrun Interrupt to INT1</p> <p>This bit allows the event to generate an interrupt on the INT1 line.</p> <p>0h (R/W) = This event will not trigger an interrupt on TX_INT1.</p> <p>1h (R/W) = A Buffer Underrun condition will trigger an interrupt on TX_INT1.</p> <p>Reset type: SYSRSn</p>
0	INT1_EN_FRAME_DONE	R/W	0h	<p>Enable Frame Done interrupt to INT1</p> <p>This bit allows the event to generate an interrupt on the INT1 line.</p> <p>0h (R/W) = This event will not trigger an interrupt on TX_INT1.</p> <p>1h (R/W) = A Frame Done event will trigger an interrupt on TX_INT1.</p> <p>Reset type: SYSRSn</p>

**28.6.3.14 TX\_DMA\_CTRL Register (Offset = 11h) [reset = 0h]**

TX\_DMA\_CTRL is shown in [Figure 28-56](#) and described in [Table 28-60](#).

Return to the [Summary Table](#).

Transmit DMA event control register

**Figure 28-56. TX\_DMA\_CTRL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							DMA_EVT_EN
R-0h							R/W-0h

**Table 28-60. TX\_DMA\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	DMA_EVT_EN	R/W	0h	<p>DMA Event Enable bit</p> <p>This bit will enable the DMA event to be generated upon the completion of a transmit frame.</p> <p>0h (R/W) = A DMA event will not be generated.</p> <p>1h (R/W) = A DMA event will be generated upon the completion of a transmitted frame.</p> <p>Note: The DMA event will only be generated for data frames.</p> <p>Reset type: SYSRSn</p>

**28.6.3.15 TX\_LOCK\_CTRL Register (Offset = 12h) [reset = 0h]**

TX\_LOCK\_CTRL is shown in [Figure 28-57](#) and described in [Table 28-61](#).

Return to the [Summary Table](#).

Transmit lock control register

**Figure 28-57. TX\_LOCK\_CTRL Register**

15	14	13	12	11	10	9	8
KEY							
W-0h							
7	6	5	4	3	2	1	0
RESERVED							LOCK
R-0h							R/W-0h

**Table 28-61. TX\_LOCK\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	KEY	W	0h	Write Key In order to write to this register, 0xA5 must be written to this field at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every change to this register. Reset type: SYSRSn
7-1	RESERVED	R	0h	Reserved
0	LOCK	R/W	0h	Control Register Lock Enable bit This bit locks the contents of all the transmit control registers that support a lock protection. Once locked, further writes will not take effect until a SYSRS has reset this register. Once set, further writes to this bit will be ignored.  0h (R/W) = Transmit control registers can be modified and are not locked. 1h (R/W) = Transmit control registers are locked and cannot be modified until this bit is cleared by SYSRS. Any further writes to this bit are ignored.  Note: The KEY field must contain 0xA5 for any write to this bit to take effect. Reset type: SYSRSn

**28.6.3.16 TX\_EVT\_STS Register (Offset = 14h) [reset = 0h]**

TX\_EVT\_STS is shown in [Figure 28-58](#) and described in [Table 28-62](#).

Return to the [Summary Table](#).

Transmit event and error status flag register

**Figure 28-58. TX\_EVT\_STS Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				PING_TRIGGE RED	BUF_OVERRU N	BUF_UNDERR UN	FRAME_DONE
R-0h				R-0h	R-0h	R-0h	R-0h

**Table 28-62. TX\_EVT\_STS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	PING_TRIGGERED	R	0h	<p>Ping Frame Triggered Flag Bit</p> <p>This bit indicates that a ping frame has been triggered. This bit is set by hardware when either the ping timer or an external trigger event have occurred. Software can also force this bit to get set by writing to the TX_EVT_FRC register.</p> <p>0h (R) = A ping frame has not been triggered. 1h (R) = A ping frame has been triggered by either the ping timer or external trigger.</p> <p>To clear this bit, write to the corresponding bit in the TX_EVT_CLR register. Reset type: SYSRSn</p>
2	BUF_OVERRUN	R	0h	<p>Buffer Overrun Flag Bit</p> <p>This bit indicates that buffer overrun has occurred. Software can also force this bit to get set by writing to the TX_EVT_FRC register.</p> <p>0h (R) = Buffer Overrun has not occurred. 1h (R) = Buffer Overrun has occurred.</p> <p>To clear this bit, write to the corresponding bit in the TX_EVT_CLR register. Reset type: SYSRSn</p>
1	BUF_UNDERRUN	R	0h	<p>Buffer Underrun Flag Bit</p> <p>This bit indicates that buffer underrun has occurred. Software can also force this bit to get set by writing to the TX_EVT_FRC register.</p> <p>0h (R) = Buffer Underrun has not occurred. 1h (R) = Buffer Underrun has occurred.</p> <p>To clear this bit, write to the corresponding bit in the TX_EVT_CLR register. Reset type: SYSRSn</p>



**Table 28-62. TX\_EVT\_STS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	FRAME_DONE	R	0h	<p>Frame Done Flag Bit</p> <p>This bit indicates a Frame Done condition. This bit is set by hardware when a frame transmission has been completed. Software can also force this bit to get set by writing to the TX_EVT_FRC register.</p> <p>0h (R) = Frame Done condition has not occurred. 1h (R) = Frame Done condition has occurred.</p> <p>To clear this bit, write to the corresponding bit in the TX_EVT_CLR register.</p> <p>Reset type: SYSRSn</p>

**28.6.3.17 TX\_EVT\_CLR Register (Offset = 16h) [reset = 0h]**

TX\_EVT\_CLR is shown in [Figure 28-59](#) and described in [Table 28-63](#).

Return to the [Summary Table](#).

Transmit event and error clear register

**Figure 28-59. TX\_EVT\_CLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				PING_TRIGGE RED	BUF_OVERRU N	BUF_UNDERR UN	FRAME_DONE
R-0h				W-0h	W-0h	W-0h	W-0h

**Table 28-63. TX\_EVT\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	PING_TRIGGERED	W	0h	Ping Frame Triggered Flag Clear bit This bit clears the corresponding bit in the TX_EVT_STS register.  0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Writing a 1 to this bit will clear the corresponding bit in the TX_EVT_STS register to 0.  Note: This bit may not always be cleared when writing to the corresponding TX_EVT_CLR bit. If PING_TIMEOUT MODE is configured to be 0, a hardware ping timeout may occur when another frame is actively being transmitted. In this case, if this bit still shows as 1 after the clear bit is written then the ping frame has been triggered but not serviced. This bit does not indicate that the ping frame has been completely sent, only that it has been triggered by the timeout event. Reset type: SYSRSn
2	BUF_OVERRUN	W	0h	Buffer Overrun Flag Clear bit This bit clears the corresponding bit in the TX_EVT_STS register.  0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Writing a 1 to this bit will clear the corresponding bit in the TX_EVT_STS register to 0. Reset type: SYSRSn
1	BUF_UNDERRUN	W	0h	Buffer Underrun Flag Clear bit This bit clears the corresponding bit in the TX_EVT_STS register.  0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Writing a 1 to this bit will clear the corresponding bit in the TX_EVT_STS register to 0. Reset type: SYSRSn
0	FRAME_DONE	W	0h	Frame Done Flag Clear bit This bit clears the corresponding bit in the TX_EVT_STS register.  0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Writing a 1 to this bit will clear the corresponding bit in the TX_EVT_STS register to 0. Reset type: SYSRSn

### 28.6.3.18 TX\_EVT\_FRC Register (Offset = 17h) [reset = 0h]

TX\_EVT\_FRC is shown in [Figure 28-60](#) and described in [Table 28-64](#).

Return to the [Summary Table](#).

Transmit event and error flag force register

**Figure 28-60. TX\_EVT\_FRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				PING_TRIGGE RED	BUF_OVERRU N	BUF_UNDERR UN	FRAME_DONE
R-0h				W-0h	W-0h	W-0h	W-0h

**Table 28-64. TX\_EVT\_FRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	PING_TRIGGERED	W	0h	<p>Ping Frame Triggered Flag Force bit</p> <p>This bit will cause the corresponding bit in the TX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding flag bit in the TX_EVT_STS Register. Reset type: SYSRSn</p>
2	BUF_OVERRUN	W	0h	<p>Buffer Overrun Flag Force bit</p> <p>This bit will cause the corresponding bit in the TX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (R/W) = Writing a 0 to this bit will have no effect. 1h (R/W) = Force the corresponding flag bit in the TX_EVT_STS Register. Reset type: SYSRSn</p>
1	BUF_UNDERRUN	W	0h	<p>Buffer Underrun Flag Force bit</p> <p>This bit will cause the corresponding bit in the TX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding flag bit in the TX_EVT_STS Register. Reset type: SYSRSn</p>
0	FRAME_DONE	W	0h	<p>Frame Done Flag Force bit</p> <p>This bit will cause the corresponding bit in the TX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.</p> <p>0h (W) = Writing a 0 to this bit will have no effect. 1h (W) = Force the corresponding flag bit in the TX_EVT_STS Register. Reset type: SYSRSn</p>

**28.6.3.19 TX\_USER\_CRC Register (Offset = 18h) [reset = 0h]**

TX\_USER\_CRC is shown in [Figure 28-61](#) and described in [Table 28-65](#).

Return to the [Summary Table](#).

Transmit user-defined CRC register

**Figure 28-61. TX\_USER\_CRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
USER_CRC							
R/W-0h							

**Table 28-65. TX\_USER\_CRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	USER_CRC	R/W	0h	User-defined CRC This register contains the 8-bit CRC value to be transmitted in the next frame if the transmission is set for user-defined CRC option (TX_OPER_CTRL_LO.SW_CRC = 1). This register is ignored if the hardware CRC generation is enabled. Reset type: SYSRSn

**28.6.3.20 TX\_ECC\_DATA Register (Offset = 20h) [reset = 0h]**

TX\_ECC\_DATA is shown in [Figure 28-62](#) and described in [Table 28-66](#).

Return to the [Summary Table](#).

Transmit ECC data register

**Figure 28-62. TX\_ECC\_DATA 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
DATA_HIGH																DATA_LOW															
R/W-0h																R/W-0h															

**Table 28-66. TX\_ECC\_DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	DATA_HIGH	R/W	0h	Upper 16 bits of ECC Data Writing to this bitfield will cause the ECC logic to compute the ECC(SEC-DED) the entire 32-bit register and update TX_ECC_VAL register with the results. Software should write to these 16 bits of the register in a 32-bit write when needing to compute ECC for 32-bits for the full TX_ECC_DATA register. Reset type: SYSRSn
15-0	DATA_LOW	R/W	0h	Lower 16 bits of ECC Data Writing to this bitfield will cause the ECC logic to compute the ECC(SEC-DED) for these 16 bits and update the TX_ECC_VAL register with the results. Software should write to these register bits as a 16-bit write when needing to compute ECC for 16-bits. Reset type: SYSRSn

**28.6.3.21 TX\_ECC\_VAL Register (Offset = 22h) [reset = Ch]**

TX\_ECC\_VAL is shown in [Figure 28-63](#) and described in [Table 28-67](#).

Return to the [Summary Table](#).

Transmit ECC value register

**Figure 28-63. TX\_ECC\_VAL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		ECC_VAL					
R-0h		R-Ch					

**Table 28-67. TX\_ECC\_VAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6-0	ECC_VAL	R	Ch	Computed ECC Value This field contains the ECC value computed using SEC-DED either for 16-bit or 32-bit data in the TX_ECC_DATA register. Reset type: SYSRSn

**28.6.3.22 TX\_BUF\_BASE\_y Register (Offset = 40h + formula) [reset = 0h]**

TX\_BUF\_BASE\_y is shown in [Figure 28-64](#) and described in [Table 28-68](#).

Return to the [Summary Table](#).

Base address for transmit buffer

Offset = 40h + (y \* 1h); where y = 0h to Fh

**Figure 28-64. TX\_BUF\_BASE\_y Register**

15	14	13	12	11	10	9	8
BASE_ADDRESS							
R/W-0h							
7	6	5	4	3	2	1	0
BASE_ADDRESS							
R/W-0h							

**Table 28-68. TX\_BUF\_BASE\_y Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	BASE_ADDRESS	R/W	0h	Transmit Data Buffer Base Address This is the base address of the 16-word data buffer used by the transmitter. Reset type: SYSRSn

## 28.7 Register to Driverlib Function Mapping

**Table 28-69. FSI Registers to Driverlib Functions**

File	Driverlib Function
<b>TX_MASTER_CTRL</b>	
fsi.c	FSI_resetTxModule
fsi.c	FSI_clearTxModuleReset
fsi.h	FSI_sendTxFlush
fsi.h	FSI_stopTxFlush
<b>TX_CLK_CTRL</b>	
fsi.c	FSI_resetTxModule
fsi.c	FSI_clearTxModuleReset
fsi.h	FSI_enableTxClock
fsi.h	FSI_disableTxClock
<b>TX_OPER_CTRL_LO</b>	
fsi.h	FSI_selectTxPLLClock
fsi.h	FSI_setTxDataWidth
fsi.h	FSI_enableTxSPIMode
fsi.h	FSI_disableTxSPIMode
fsi.h	FSI_setTxStartMode
fsi.h	FSI_setTxPingTimeoutMode
fsi.h	FSI_enableTxUserCRC
fsi.h	FSI_disableTxUserCRC
<b>TX_OPER_CTRL_HI</b>	
fsi.h	FSI_setTxExtFrameTrigger
fsi.h	FSI_enableTxCRCForceError
fsi.h	FSI_disableTxCRCForceError
fsi.h	FSI_setTxECCComputeWidth
<b>TX_FRAME_CTRL</b>	
fsi.h	FSI_setTxFrameType
fsi.h	FSI_setTxSoftwareFrameSize
fsi.h	FSI_startTxTransmit
<b>TX_FRAME_TAG_UDATA</b>	
fsi.h	FSI_setTxFrameTag
fsi.h	FSI_setTxUserDefinedData
<b>TX_BUF_PTR_LOAD</b>	
fsi.h	FSI_setTxBufferPtr
<b>TX_BUF_PTR_STS</b>	
fsi.h	FSI_getTxBufferPtr
fsi.h	FSI_getTxWordCount
<b>TX_PING_CTRL</b>	
fsi.c	FSI_resetTxModule
fsi.c	FSI_clearTxModuleReset
fsi.h	FSI_enableTxPingTimer
fsi.h	FSI_disableTxPingTimer
fsi.h	FSI_enableTxExtPingTrigger
fsi.h	FSI_disableTxExtPingTrigger
<b>TX_PING_TAG</b>	
fsi.h	FSI_enableTxPingTimer



**Table 28-69. FSI Registers to Driverlib Functions (continued)**

File	Driverlib Function
fsi.h	FSI_setTxPingTag
<b>TX_PING_TO_REF</b>	
fsi.h	FSI_enableTxPingTimer
<b>TX_PING_TO_CNT</b>	
fsi.h	FSI_getTxCurrentPingTimeoutCounter
<b>TX_INT_CTRL</b>	
fsi.h	FSI_enableTxInterrupt
fsi.h	FSI_disableTxInterrupt
<b>TX_DMA_CTRL</b>	
fsi.h	FSI_enableTxDMAEvent
fsi.h	FSI_disableTxDMAEvent
<b>TX_LOCK_CTRL</b>	
fsi.h	FSI_lockTxCtrl
<b>TX_EVT_STS</b>	
fsi.h	FSI_getTxEventStatus
<b>TX_EVT_CLR</b>	
fsi.h	FSI_clearTxEvents
<b>TX_EVT_FRC</b>	
fsi.h	FSI_forceTxEvents
<b>TX_USER_CRC</b>	
fsi.h	FSI_enableTxUserCRC
<b>TX_ECC_DATA</b>	
fsi.h	FSI_setTxECCdata
<b>TX_ECC_VAL</b>	
fsi.h	FSI_getTxECCValue
<b>TX_BUF_BASE</b>	
fsi.c	FSI_writeTxBuffer
fsi.h	FSI_getTxBufferAddress
<b>RX_MASTER_CTRL</b>	
fsi.c	FSI_resetRxModule
fsi.c	FSI_clearRxModuleReset
fsi.h	FSI_enableRxInternalLoopback
fsi.h	FSI_disableRxInternalLoopback
fsi.h	FSI_enableRxSPIPairing
fsi.h	FSI_disableRxSPIPairing
<b>RX_OPER_CTRL</b>	
fsi.h	FSI_setRxDataWidth
fsi.h	FSI_enableRxSPIMode
fsi.h	FSI_disableRxSPIMode
fsi.h	FSI_setRxSoftwareFrameSize
fsi.h	FSI_setRxECCComputeWidth
fsi.h	FSI_setRxPingTimeoutMode
<b>RX_FRAME_INFO</b>	
fsi.h	FSI_getRxFrameType
<b>RX_FRAME_TAG_UDATA</b>	
fsi.h	FSI_getRxFrameTag
fsi.h	FSI_getRxUserDefinedData

**Table 28-69. FSI Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>RX_DMA_CTRL</b>	
fsi.h	FSI_enableRxDMAEvent
fsi.h	FSI_disableRxDMAEvent
<b>RX_EVT_STS</b>	
fsi.h	FSI_getRxEventStatus
<b>RX_CRC_INFO</b>	
fsi.h	FSI_getRxReceivedCRC
fsi.h	FSI_getRxComputedCRC
<b>RX_EVT_CLR</b>	
fsi.h	FSI_clearRxEvents
<b>RX_EVT_FRC</b>	
fsi.h	FSI_forceRxEvents
<b>RX_BUF_PTR_LOAD</b>	
fsi.h	FSI_setRxBufferPtr
<b>RX_BUF_PTR_STS</b>	
fsi.h	FSI_getRxBufferPtr
fsi.h	FSI_getRxWordCount
<b>RX_FRAME_WD_CTRL</b>	
fsi.c	FSI_resetRxModule
fsi.c	FSI_clearRxModuleReset
fsi.h	FSI_enableRxFrameWatchdog
fsi.h	FSI_disableRxFrameWatchdog
<b>RX_FRAME_WD_REF</b>	
fsi.h	FSI_enableRxFrameWatchdog
<b>RX_FRAME_WD_CNT</b>	
fsi.h	FSI_getRxFrameWatchdogCounter
<b>RX_PING_WD_CTRL</b>	
fsi.c	FSI_resetRxModule
fsi.c	FSI_clearRxModuleReset
fsi.h	FSI_enableRxPingWatchdog
fsi.h	FSI_disableRxPingWatchdog
<b>RX_PING_TAG</b>	
fsi.h	FSI_getRxPingTag
<b>RX_PING_WD_REF</b>	
fsi.h	FSI_enableRxPingWatchdog
<b>RX_PING_WD_CNT</b>	
fsi.h	FSI_getRxPingWatchdogCounter
<b>RX_INT1_CTRL</b>	
fsi.h	FSI_enableRxInterrupt
fsi.h	FSI_disableRxInterrupt
<b>RX_INT2_CTRL</b>	
fsi.h	FSI_enableRxInterrupt
fsi.h	FSI_disableRxInterrupt
<b>RX_LOCK_CTRL</b>	
fsi.h	FSI_lockRxCtrl
<b>RX_ECC_DATA</b>	
fsi.h	FSI_setRxECCData

**Table 28-69. FSI Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>RX_ECC_VAL</b>	
fsi.h	FSI_setRxReceivedECCValue
<b>RX_ECC_SEC_DATA</b>	
fsi.h	FSI_getRxECCCorrectedData
<b>RX_ECC_LOG</b>	
fsi.h	FSI_getRxECCLog
<b>RX_DLYLINE_CTRL</b>	
fsi.c	FSI_configRxDelayLine
<b>RX_BUF_BASE</b>	
fsi.c	FSI_readRxBuffer
fsi.h	FSI_getRxBufferAddress

## Configurable Logic Block (CLB)

---

---

This chapter describes the features and operation of the C2000 configurable logic block (CLB) which is a collection of configurable blocks that may be inter-connected using software to implement custom digital logic functions.

Topic	Page
<b>29.1 Introduction</b> .....	<b>2685</b>
<b>29.2 Features</b> .....	<b>2685</b>
<b>29.3 CLB Input/Output Connection</b> .....	<b>2686</b>
<b>29.4 The CLB Tile</b> .....	<b>2693</b>
<b>29.5 CPU Interface</b> .....	<b>2703</b>
<b>29.6 CLB Registers</b> .....	<b>2705</b>

## 29.1 Introduction

The C2000 configurable logic block (CLB) is a collection of configurable blocks that can be interconnected using software to implement custom digital logic functions. The CLB is able to enhance existing peripherals through a set of crossbar interconnections, which provides a high level of connectivity to existing control peripherals such as enhanced pulse width modulators (ePWM), enhanced capture modules (eCAP), and enhanced quadrature encoder pulse modules (eQEP). The crossbars also allow the CLB to be connected to external GPIO pins. In this way, the CLB can be configured to interact with device peripherals to perform small logical functions such as simple PWM generators, or to implement custom serial data exchange protocols.

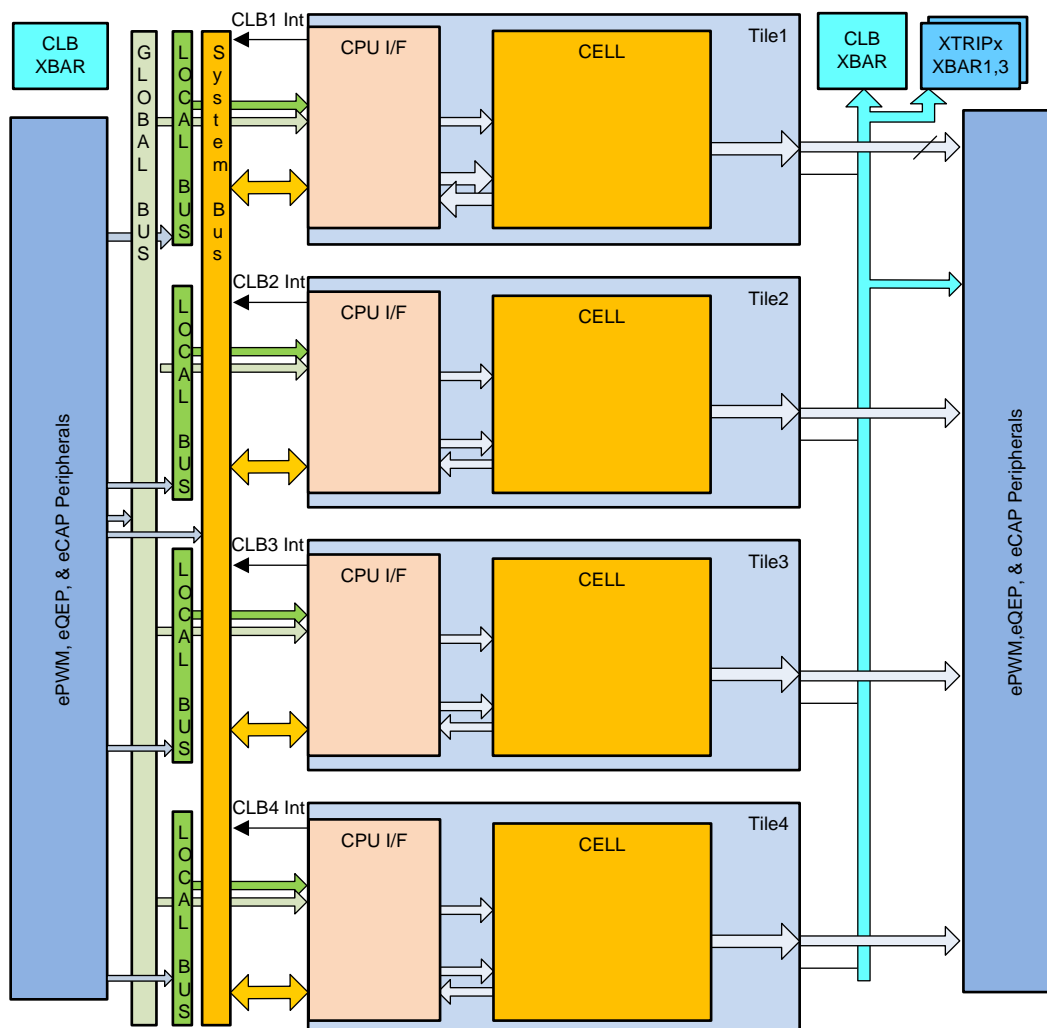
The CLB peripheral is configured through the CLB Tool. For more information on the CLB tool, available examples, application reports and user guide, please refer to the following location in your C2000WARE package (C2000Ware\_2\_00\_00\_03 and higher):

**C2000WARE\_INSTALL\_LOCATION\utilities\clb\_tool\clb\_syscfg\doc**

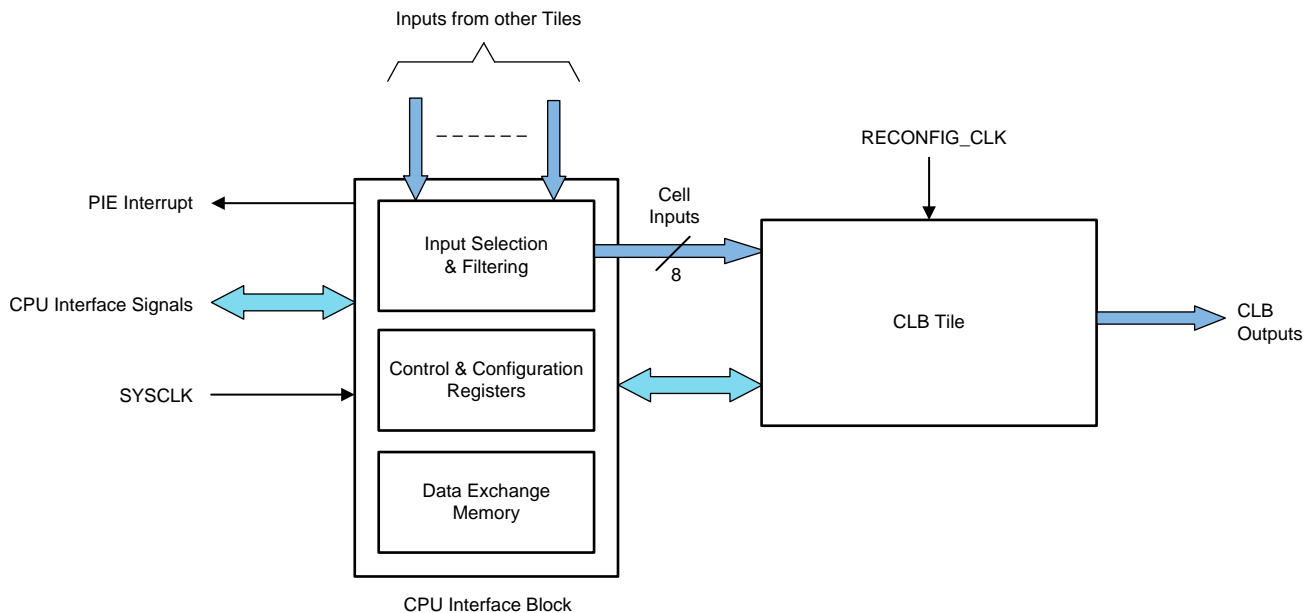
## 29.2 Features

The CLB subsystem contains a number of identical tiles. There are four such tiles in the CLB subsystem; other devices may contain more or fewer tiles. Tiles are numbered 1 to 'N', where 'N' is the total tile count on the device. Each tile contains combinational and sequential logic blocks, as well as other dedicated hardware to be described later in this document. The figure below shows the structure of the CLB subsystem in the device.

**Figure 29-1. Block Diagram of the CLB Subsystem in the Device**



The tile contains the core logic providing the logic reconfiguration capability. Each CLB tile is associated with a separate CPU interface, which contains the registers needed to control and configure the logic in the tile. The CPU interface also contains data transfer buffers that can be used as part of the configurable logic to exchange data with the rest of the device. The figure below shows the connections between the tile, the CPU interface, and the device.

**Figure 29-2. Block Diagram of a CLB Tile and CPU Interface**


## 29.3 CLB Input/Output Connection

### 29.3.1 Overview

There are four instances of the CLB module in the device. Each CLB instance sees a common set 72 input signals referred to as global input signals. Additionally, each CLB instance has a specific set of 25 input signals that are unique to each instance, and are referred to as local input signals. Each of the eight inputs of a CLB can be chosen from any of the global input signals or the local input signals.

### 29.3.2 CLB Input Selection

Each CLB module has eight inputs that feed into the reconfigurable logic cell. Each of these inputs can be selectively driven by a predefined set of signals. A two-level mux structure allows each input of each CLB instance to select a signal.

A set of 72 signals is common to all the CLB instances. These are referred to as global inputs in the figure below. A separate set of 25 signals is unique to each instance of the CLB. These are referred to as local inputs in the figure below.

Registers `CLB_LCL_MUX_SEL_1` and `CLB_LCL_MUX_SEL_2` control the local mux selection for each of the eight inputs. The mux control registers `CLB_GLBL_MUX_SEL_1` and `CLB_GLBL_MUX_SEL_2` control the global mux selection for each of the eight inputs.

The local mux select value of 0 causes the selected global mux input signal to be connected to the corresponding CLB Input. For example, setting `CLB_LCL_MUX_SEL_IN_0 = 0` and `CLB_GLBL_MUX_SEL_IN_0 = 8` causes the global mux input number 8 to be connected to CLB Input 0.

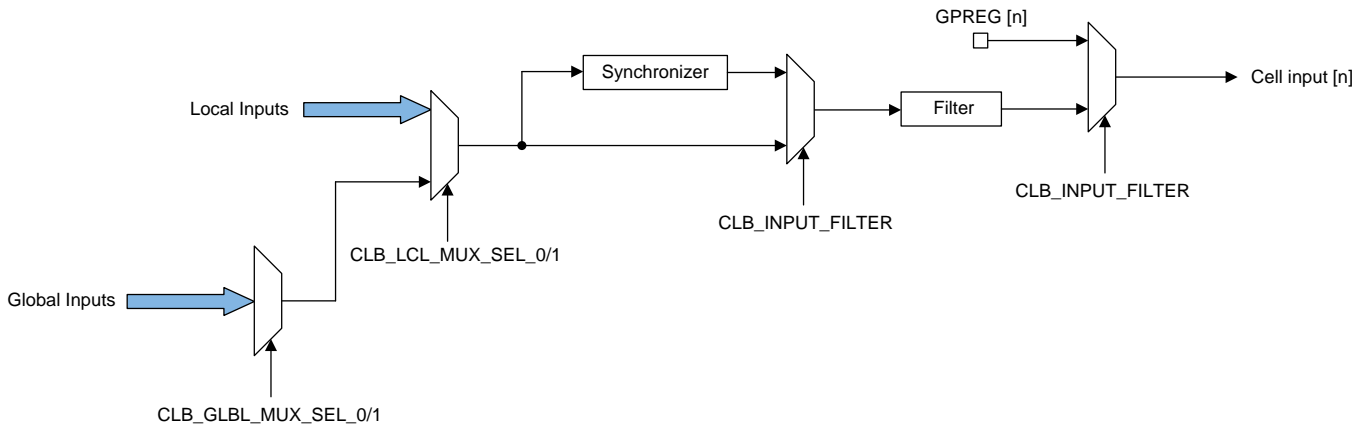


Figure 29-3. CLB Input Mux and Filter

The global and local input mux settings are shown in [Table 29-1](#) and [Table 29-2](#).

Table 29-1. Global Signals and Mux Selection

Global Input Mux Bit Position	Signal Name	Instance Name
0	ePWMxA	EPWM1
1	PWMA[OE] <sup>(1)</sup>	EPWM1
2	ePWMxB	EPWM1
3	PWMB[OE] <sup>(1)</sup>	EPWM1
4	CTR=ZERO	EPWM1
5	CTR=PRD	EPWM1
6	CTR_Dir	EPWM1
7	TBCLK	EPWM1
8	CTR=CMPA	EPWM1
9	CTR=CMPB	EPWM1
10	CTR=CMPC	EPWM1
11	CTR=CMPD	EPWM1
12	PWMA[AQ] <sup>(2)</sup>	EPWM1
13	PWMB[AQ] <sup>(2)</sup>	EPWM1
14	PWMA[DB] <sup>(3)</sup>	EPWM1
15	PWMB[DB] <sup>(3)</sup>	EPWM1
16	ePWMxA	EPWM2
17	PWMA[OE] <sup>(1)</sup>	EPWM2
18	ePWMxB	EPWM2
19	PWMB[OE] <sup>(1)</sup>	EPWM2
20	CTR=ZERO	EPWM2
21	CTR=PRD	EPWM2
22	CTR_Dir	EPWM2
23	TBCLK	EPWM2
24	CTR=CMPA	EPWM2
25	CTR=CMPB	EPWM2
26	CTR=CMPC	EPWM2
27	CTR=CMPD	EPWM2
28	PWMA[AQ] <sup>(2)</sup>	EPWM2
29	PWMB[AQ] <sup>(2)</sup>	EPWM2
30	PWMA[DB] <sup>(3)</sup>	EPWM2

**Table 29-1. Global Signals and Mux Selection (continued)**

Global Input Mux Bit Position	Signal Name	Instance Name
31	PWMB[DB] <sup>(3)</sup>	EPWM2
32	ePWMxA	EPWM3
33	PWMA[OE] <sup>(1)</sup>	EPWM3
34	ePWMxB	EPWM3
35	PWMB[OE] <sup>(1)</sup>	EPWM3
36	CTR=ZERO	EPWM3
37	CTR=PRD	EPWM3
38	CTR_Dir	EPWM3
39	TBCLK	EPWM3
40	CTR=CMPA	EPWM3
41	CTR=CMPB	EPWM3
42	CTR=CMPC	EPWM3
43	CTR=CMPD	EPWM3
44	PWMA[AQ] <sup>(2)</sup>	EPWM3
45	PWMB[AQ] <sup>(2)</sup>	EPWM3
46	PWMA[DB] <sup>(3)</sup>	EPWM3
47	PWMB[DB] <sup>(3)</sup>	EPWM3
48	ePWMxA	EPWM4
49	PWMA[OE] <sup>(1)</sup>	EPWM4
50	ePWMxB	EPWM4
51	PWMB[OE] <sup>(1)</sup>	EPWM4
52	CTR=ZERO	EPWM4
53	CTR=PRD	EPWM4
54	CTR_Dir	EPWM4
55	TBCLK	EPWM4
56	CTR=CMPA	EPWM4
57	CTR=CMPB	EPWM4
58	CTR=CMPC	EPWM4
59	CTR=CMPD	EPWM4
60	PWMA[AQ] <sup>(2)</sup>	EPWM4
61	PWMB[AQ] <sup>(2)</sup>	EPWM4
62	PWMA[DB] <sup>(3)</sup>	EPWM4
63	PWMB[DB] <sup>(3)</sup>	EPWM4
64	AUXSIG0	CLB X BAR
65	AUXSIG1	CLB X BAR
66	AUXSIG2	CLB X BAR
67	AUXSIG2	CLB X BAR
68	AUXSIG4	CLB X BAR
69	AUXSIG5	CLB X BAR
70	AUXSIG6	CLB X BAR
71	AUXSIG7	CLB X BAR

(1) PWMA[OE] and PWMB[OE] refer to trip outputs from the respective PWM module.

(2) PWMA[AQ] and PWMB[AQ] refer to the output of the AQ submodule in the respective PWM module.

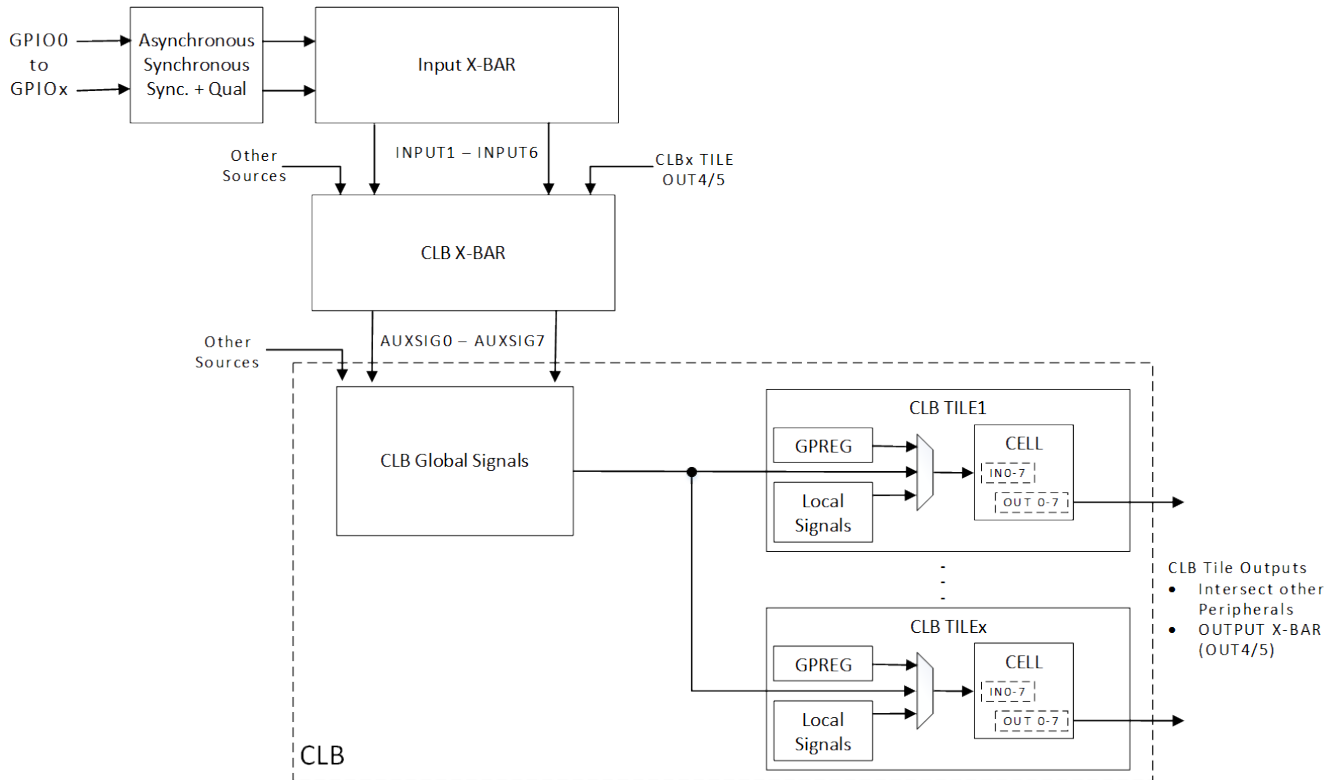
(3) PWMA[DB] and PWMB[DB] refer to the output of the DB submodule in the respective PWM module.



**Table 29-2. Local Signals and Mux Selection**

Bit Position	Signal Name	Instance Name (for CLB1)	Instance Name (for CLB2)	Instance Name (for CLB3)	Instance Name (for CLB4)
0	Global Mux input	Global Mux input	Global Mux input	Global Mux input	Global Mux input
1	DCAEVT1	EPWM1	EPWM2	EPWM3	EPWM4
2	DCAEVT2	EPWM1	EPWM2	EPWM3	EPWM4
3	DCBEVT1	EPWM1	EPWM2	EPWM3	EPWM4
4	DCBEVT2	EPWM1	EPWM2	EPWM3	EPWM4
5	DCAH	EPWM1	EPWM2	EPWM3	EPWM4
6	DCAL	EPWM1	EPWM2	EPWM3	EPWM4
7	DCBH	EPWM1	EPWM2	EPWM3	EPWM4
8	DCBL	EPWM1	EPWM2	EPWM3	EPWM4
9	OST	EPWM1	EPWM2	EPWM3	EPWM4
10	CBC	EPWM1	EPWM2	EPWM3	EPWM4
11	ECAPIN	ECAP1	ECAP2	ECAP3	ECAP4
12	ECAP_OUT	ECAP1	ECAP2	ECAP3	ECAP4
13	ECAP_OUT_EN	ECAP1	ECAP2	ECAP3	ECAP4
14	CEVT1	ECAP1	ECAP2	ECAP3	ECAP4
15	CEVT2	ECAP1	ECAP2	ECAP3	ECAP4
16	CEVT3	ECAP1	ECAP2	ECAP3	ECAP4
17	CEVT4	ECAP1	ECAP2	ECAP3	ECAP4
18	EQEPA	EQEP1	EQEP2		
19	EQEPB	EQEP1	EQEP2		
20	EQEPI	EQEP1	EQEP2		
21	EQEPS	EQEP1	EQEP2		
22	CPU1.TBCLKSYNC	CPU1.TBCLKSYNC	CPU1.TBCLKSYNC		
24	CPU1.HALT	CPU1.HALT	CPU1.HALT		

In order to use GPIOs as inputs to the CLB, the user must utilize the Input X-BAR and the CLB X-BAR. The figure below shows how GPIOs can be used as inputs to the CLB tiles.

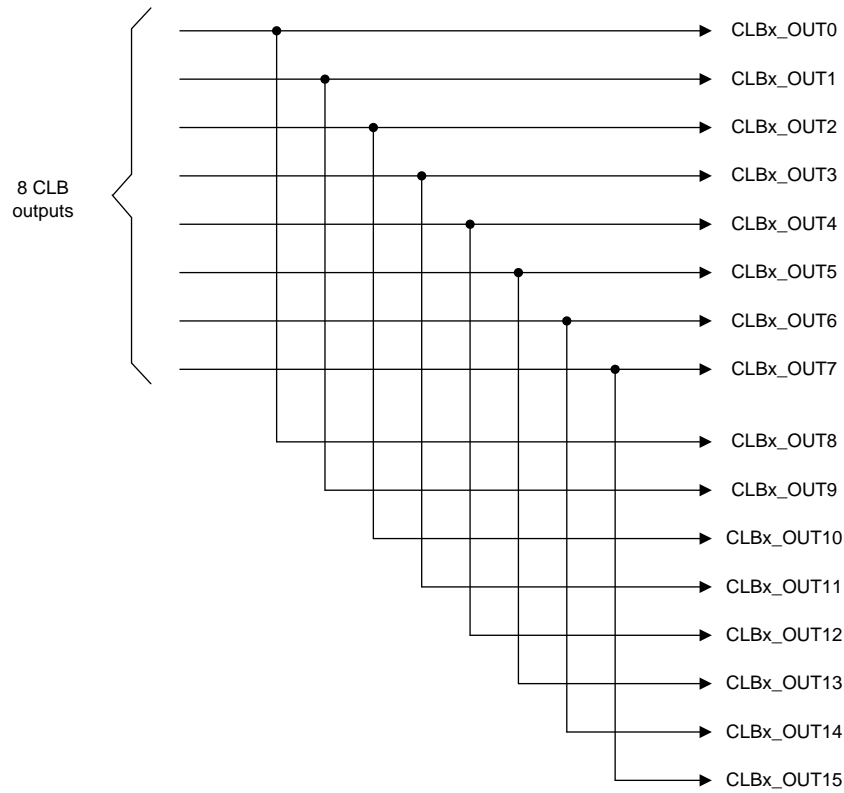


**Figure 29-4. GPIO to CLB Tile Connections**

### 29.3.3 CLB Output Selection

The eight outputs of the CLB are replicated to create 16 output signals. Each of these 16 outputs has a separate enable bit defined in the CLB output enable register, CLB\_OUT\_EN\_REG. The CLB outputs go to the ePWM, eCAP, eQEP and the crossbar module in the device. This allows the user to enhance the functionality of these modules with the CLB.

Figure 29-5. CLB Outputs



### 29.3.4 Peripheral Signal Multiplexer

Each CLB output signal passes through an external multiplexer which intersects a specific peripheral signal. The output of the multiplexer is connected to the destination of the original peripheral signal and the default multiplexer setting is that the peripheral signal is passed through. The multiplexer is controlled by a bit in the CLB output enable register CLB\_OEN.

The table below shows the allocation of peripheral signals and the CLB outputs with which they are multiplexed.

Table 29-3. Peripheral Signal Multiplexer Table

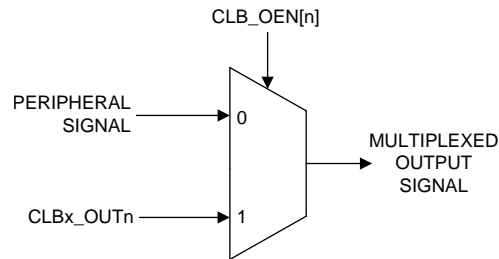
CLB Instance	CLB Output Signal	Peripheral Signal	Peripheral Name
CLB1	CLB1_OUT0_0	PWMA	EPWM1
CLB1	CLB1_OUT1_0	PWMA_OE	EPWM1
CLB1	CLB1_OUT2_0	PWMB	EPWM1
CLB1	CLB1_OUT3_0	PWMB_OE	EPWM1
CLB1	CLB1_OUT4_0	AQ_PWMA	EPWM1
CLB1	CLB1_OUT5_0	AQ_PWMB	EPWM1
CLB1	CLB1_OUT6_0	DB_PWMA	EPWM1
CLB1	CLB1_OUT7_0	DB_PWMB	EPWM1
CLB1	CLB1_OUT0_1	QA	EQEP1
CLB1	CLB1_OUT1_1	QB	EQEP1
CLB1	CLB1_OUT2_1	QDIR	EQEP1
CLB1	CLB1_OUT3_1	QCLK	EQEP1
CLB1	CLB1_OUT4_1	G1.2	CLB X BAR
CLB1	CLB1_OUT5_1	G3.2	CLB X BAR

**Table 29-3. Peripheral Signal Multiplexer Table (continued)**

CLB1	CLB1_OUT6_1	ECAPOUT	ECAP1
CLB1	CLB1_OUT7_1	ECAPOUT_EN	ECAP1
CLB2	CLB2_OUT0_0	PWMA	EPWM2
CLB2	CLB2_OUT1_0	PWMA_OE	EPWM2
CLB2	CLB2_OUT2_0	PWMB	EPWM2
CLB2	CLB2_OUT3_0	PWMB_OE	EPWM2
CLB2	CLB2_OUT4_0	AQ_PWMA	EPWM2
CLB2	CLB2_OUT5_0	AQ_PWMB	EPWM2
CLB2	CLB2_OUT6_0	DB_PWMA	EPWM2
CLB2	CLB2_OUT7_0	DB_PWMB	EPWM2
CLB2	CLB2_OUT0_1	QA	EQEP2
CLB2	CLB2_OUT1_1	QB	EQEP2
CLB2	CLB2_OUT2_1	QDIR	EQEP2
CLB2	CLB2_OUT3_1	QCLK	EQEP2
CLB2	CLB2_OUT4_1	G5.2	CLB X BAR
CLB2	CLB2_OUT5_1	G7.2	CLB X BAR
CLB2	CLB2_OUT6_1	ECAPOUT	ECAP2
CLB2	CLB2_OUT7_1	ECAPOUT_EN	ECAP2
CLB3	CLB3_OUT0_0	PWMA	EPWM3
CLB3	CLB3_OUT1_0	PWMA_OE	EPWM3
CLB3	CLB3_OUT2_0	PWMB	EPWM3
CLB3	CLB3_OUT3_0	PWMB_OE	EPWM3
CLB3	CLB3_OUT4_0	AQ_PWMA	EPWM3
CLB3	CLB3_OUT5_0	AQ_PWMB	EPWM3
CLB3	CLB3_OUT6_0	DB_PWMA	EPWM3
CLB3	CLB3_OUT7_0	DB_PWMB	EPWM3
CLB3	CLB3_OUT0_1	QA	
CLB3	CLB3_OUT1_1	QB	
CLB3	CLB3_OUT2_1	QDIR	
CLB3	CLB3_OUT3_1	QCLK	
CLB3	CLB3_OUT4_1	G9.2	CLB X BAR
CLB3	CLB3_OUT5_1	G11.2	CLB X BAR
CLB3	CLB3_OUT6_1	ECAPOUT	ECAP3
CLB3	CLB3_OUT7_1	ECAPOUT_EN	ECAP3
CLB4	CLB4_OUT0_0	PWMA	EPWM4
CLB4	CLB4_OUT1_0	PWMA_OE	EPWM4
CLB4	CLB4_OUT2_0	PWMB	EPWM4
CLB4	CLB4_OUT3_0	PWMB_OE	EPWM4
CLB4	CLB4_OUT4_0	AQ_PWMA	EPWM4
CLB4	CLB4_OUT5_0	AQ_PWMB	EPWM4
CLB4	CLB4_OUT6_0	DB_PWMA	EPWM4
CLB4	CLB4_OUT7_0	DB_PWMB	EPWM4
CLB4	CLB4_OUT0_1	QA	
CLB4	CLB4_OUT1_1	QB	
CLB4	CLB4_OUT2_1	QDIR	
CLB4	CLB4_OUT3_1	QCLK	
CLB4	CLB4_OUT4_1	G13.2	CLB X BAR

**Table 29-3. Peripheral Signal Multiplexer Table (continued)**

CLB4	CLB4_OUT5_1	G15.2	CLB X BAR
CLB4	CLB4_OUT6_1	ECAPOUT	ECAP4
CLB4	CLB4_OUT7_1	ECAPOUT_EN	ECAP4

**Figure 29-6. Peripheral Signal Multiplexer**


## 29.4 The CLB Tile

The purpose of the CLB tile is to provide the logic reconfiguration capability of the CLB. The CLB tile contains the following submodules:

- **Counter**

The counter submodule can be configured either as an adder, a counter, or a shifter. When functioning as an adder, it can either add or subtract. When functioning as a counter, it can count up or count down. When functioning as a shifter, it can shift left or shift right. The counter event inputs, as well as the reset input, may be freely connected to any of the other submodules in the same tile. There are three counters in each tile.

- **LUT4**

The LUT4 submodule has a 4-input look-up table functionality and is capable of realizing any combinatorial Boolean equation of up to four inputs. There are three LUT4 submodules in each CLB tile.

- **FSM**

The Finite State Machine (FSM) submodule can be configured either as a single four-state finite state machine, or as two independent two-state finite state machines. The FSM accepts two external inputs, and generates two state outputs and one combinational output. When not used as a state machine, the FSM submodule can accept two external inputs and function as a 4-input LUT. There are three FSM submodules in each CLB tile.

- **Output LUT**

The output LUT is a 3-input lookup table submodule capable of realizing any combinatorial Boolean equation of up to three inputs. There are eight such blocks in a CLB tile, each associated with one of the tile outputs.

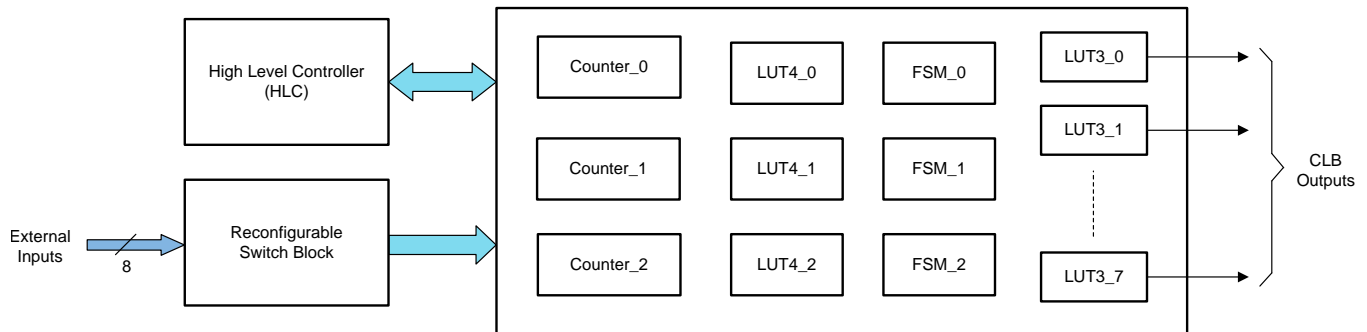
- **High Level Controller**

The High Level Controller (HLC) submodule is an event-driven block which can handle up to four concurrent events. The event can be an activity on any of the other block outputs. A predefined set of operations is executed when each event occurs. The HLC also provides a data exchange and interrupt mechanism to the CPU subsystem. There are four working registers (R0, R1, R2, and R3) which can be used for basic operations, and to modify or set up values for the three counter blocks. Unlike the other submodules, there is only one HLC in each CLB tile.

- **Configurable Switch Block**

The configurable switch block provides dynamic connectivity between all of the blocks listed above. submodules can be connected by the user, with the sole restriction that they must not form a combinational loop within the tile.

A CLB tile consists of three sets each of the counter block, FSM, and LUT4, one high-level controller, and eight output LUT blocks. The submodule numbering is shown in [Figure 29-7](#).



**Figure 29-7. The CLB Tile Submodules**

The functionality of the LUT sub-modules is configured using a register field containing the binary pattern of the output of the desired look-up table. For example, a 4-input LUT has 16 possible input permutations, each of which corresponds to a desired binary 0 or 1 at its output. The register field would therefore be 16-bits in length, with each bit representing the desired result of a binary pattern. Input pattern sequences start at 0000 and continue sequentially to 1111. A similar method is used to encode the 16-bit state equations in the FSM sub-module.

### 29.4.1 Static Switch Block

The Static switch Block provides the configurable connectivity between the sub-modules in the CLB tile. The outputs of all the sub-modules and the eight external inputs are connected to a common internal bus inside the tile. Every input port has a 32-to-1 multiplexer and an associated 5-bit selection value which allows the user to select one of the inputs on the bus. The only restrictions are certain signals (described below) which are tied off in the design to prevent creation of accidental combinatorial loops.

**Table 29-4. Output Table**

Bit position	Signal connection	Comment
0	Always '0'	This select value will be used to tie an input to '0'.
1	COUNTER_0 MATCH2	
2	COUNTER_0 ZERO	
3	COUNTER_0 MATCH1	
4	FSM_0 STATE_BIT_0	
5	FSM_0 STATE_BIT_1	
6	FSM_0 LUT output	
7	LUT4_0 output	
8	Always '1'	This select value will be used to tie an input to '1'.
9	COUNTER_1 MATCH2	
10	COUNTER_1 ZERO	
11	COUNTER_1 MATCH1	
12	FSM_1 STATE_BIT_0	
13	FSM_1 STATE_BIT_1	
14	FSM_1 LUT output	
15	LUT4_1 output	
16	Always '0'	
17	COUNTER_2 MATCH2	
18	COUNTER_2 ZERO	
19	COUNTER_2 MATCH1	

**Table 29-4. Output Table (continued)**

20	FSM_2 STATE_BIT_0	
21	FSM_2 STATE_BIT_1	
22	FSM_2 LUT output	
23	LUT4_2 output	
24	External Input 0	
25	External Input 1	
26	External Input 2	
27	External Input 3	
28	External Input 4	
29	External Input 5	
30	External Input 6	
31	External Input 7	

**Table 29-5. Input Table**

Module name	Port name	Description
Counter Block	RESET	Acts as an active high reset when used as a counter
	MODE_0	Acts as a enable when used as a counter. The counter will count only when this input is '1'.
	MODE_1	Acts as a direction control when used as a counter. If this input is '1', then the counter counts up else, it counts down.
LUT	IN0	Input 0 of the 4-input LUT.
	IN1	Input 1 of the 4-input LUT.
	IN2	Input 2 of the 4-input LUT.
	IN3	Input 3 of the 4-input LUT.
FSM	EXT_IN0	Input 0 of the FSM block.
	EXT_IN1	Input 1 of the FSM block.
	EXTRA_EXT_IN0	Extra external Input 0 of the FSM block. This input matters only if it is configured in the LUT mode.
	EXTRA_EXT_IN1	Extra external Input 1 of the FSM block. This input matters only if it is configured in the LUT mode.

The static switch block allows the user to define the input connection of any sub-module to come from any of the outputs in the output table, above. It is therefore easy to create a combinatorial loop. In order to prevent this, certain paths are broken in the input path of each sub-module. These port positions are tied to '0', as shown in the table below.

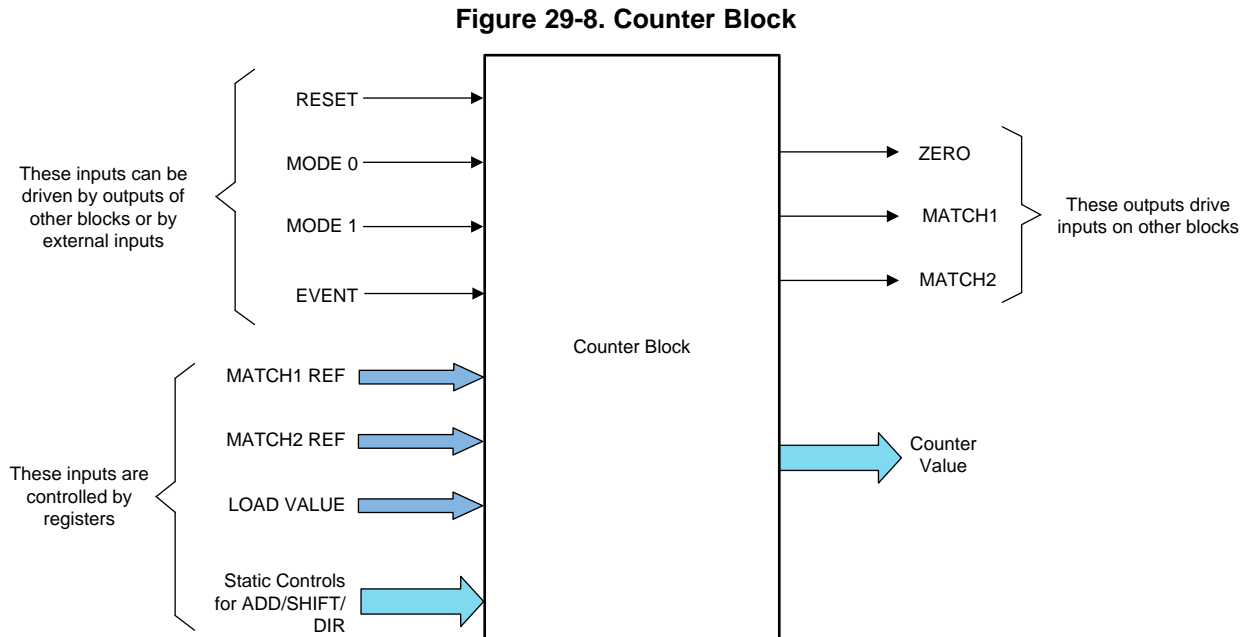
**Table 29-6. Ports Tied Off to Prevent Combinatorial Loops**

Module name	Ports of input MUX tied off to '0' to prevent combinatorial loops
LUT_0	LUT_0 , LUT_1 and LUT_2 output, FSM_0, FSM_1 and FSM_2 output.
FSM_0	LUT_1 and LUT_2 output, FSM_0, FSM_1 and FSM_2 output.
LUT_1	LUT_1 and LUT_2 output, FSM_1 and FSM_2 output.
FSM_1	LUT_2 output, FSM_1 and FSM_2 output.
LUT_2	LUT_2 output, FSM_2 output.
FSM_2	FSM_2 output.

## 29.4.2 Counter Block

### 29.4.2.1 Counter Description

The counter block is a complex functional submodule that can be configured either as a counter, an adder, or a shifter. Apart from the normal operational control, this block has a dedicated EVENT input, which can trigger an addition, subtraction or shift operation, or load data into the counter register. The inputs to the counter sub are shown in the figure below.



### 29.4.2.2 Counter Operation

At the heart of the counter block is a 32-bit count register. This register can either be loaded statically before counting commences, or dynamically at run time. The operation of the counter submodule is determined by the inputs described below. Note that each of the inputs can be connected to the outputs of any of the other blocks in the CLB tile. These connections are made by configuring the configurable switch block.

The counter inputs are as follows.

- **RESET:** This is the highest priority input and takes precedence over all other inputs. The input is level sensitive and as long as it remains high, the counter will reset to 0 on the next clock cycle.
- **MODE\_0:** This input is an enable for the counter. The counter begins counting (up or down depending on the MODE\_1 setting) only when this input is high. If this input is low, then no counting takes place.
- **MODE\_1:** This input is the direction control for the counter. If this input is high, then the counter will increment on every clock cycle where it sees MODE\_0 to be high. If this input is low, then the counter decrements for every clock cycle it sees MODE\_0 high. The counter wraps around to 0x00000000 after 0xFFFFFFFF when counting up. The counter wraps around to 0xFFFFFFFF after 0x00000000 when counting down. The only exception to this is when an EVENT occurs at exactly the same time, causing a different value to be loaded into the counter.
- **EVENT:** This input is defined for the purpose of triggering actions in the counter based on certain events. The event itself can be any of the outputs of the other blocks or an external input to the tile. The counter's static control inputs define the behavior of the counter on an active event. An active event is defined as a rising edge on the EVENT input. The counter can be configured to perform one of the following actions:
  - Load a predefined 32-bit value from the LOAD VALUE register into the count register
  - Shift the contents of the counter register left or right by a predefined amount between 0 and 31
  - Add or subtract a predefined 32-bit value. Addition and subtraction are treated as 32-bit unsigned



operations and there is no saturation.

Note that the effect of a rising edge on the EVENT input only lasts for one cycle. On the next cycle, the counter operation will continue based on the MODE\_0, MODE\_1, and RESET inputs.

MATCH1 REF and MATCH2 REF are 32-bit reference values that are used to generate the MATCH1 and MATCH2 outputs. The MATCH1 output becomes active high whenever the counter register value matches the 32-bit MATCH1 REF value. MATCH2 behaves in a similar manner in relation to the MATCH2 REF register. The reference values for MATCH1 and MATCH2 can either be setup once before the start of operation, or can be modified dynamically. The High Level Controller can load desired values into the MATCH1 REF and MATCH2 REF registers.

Note that the counter load and match registers are not memory-mapped. For more information, see [Table 29-12](#).

The three logic outputs of the counter block are as follows:

- **ZERO:** This output goes high whenever the counter register is zero.
- **MATCH1:** This output goes high whenever the counter register is equal to the MATCH1 REF input register.
- **MATCH2:** This output goes high whenever the counter register is equal to the MATCH2 REF input register.

The operation of the counter block is controlled by the CFG\_MISC\_CTRL register. The following three bits of this register are relevant for each counter. The “x” below refers to the counter instance; 0, 1, or 2. For more information, see the CLB\_MISC\_CONTROL register description located in [Section 29.6](#).

- COUNT\_EVENT\_CTRL\_x: This bit defines whether the counter performs an addition or a shift on an event. A value of 0 means that on an event, the counter will just load the static value; 1 means and add/shift operation will be performed.
- COUNT\_ADD\_SHIFT\_x. 1 means add, 0 means shift.
- COUNT\_DIR\_x. 1 means left shift or add. 0 means right shift or subtract.

The table below shows the logical operation of the counter block in terms of the inputs and control register bits. Count up and down modes are the normal operation with EVENT = 0. The operations on the CNTVAL register are as follows:

Load:  $CNTVAL = EVENT\_LOAD\_VAL$

Shift right:  $CNTVAL = CNTVAL \gg EVENT\_LOAD\_VAL$

Shift left:  $CNTVAL = CNTVAL \ll EVENT\_LOAD\_VAL$

Subtract:  $CNTVAL = CNTVAL - EVENT\_LOAD\_VAL$

Add:  $CNTVAL = CNTVAL + EVENT\_LOAD\_VAL$

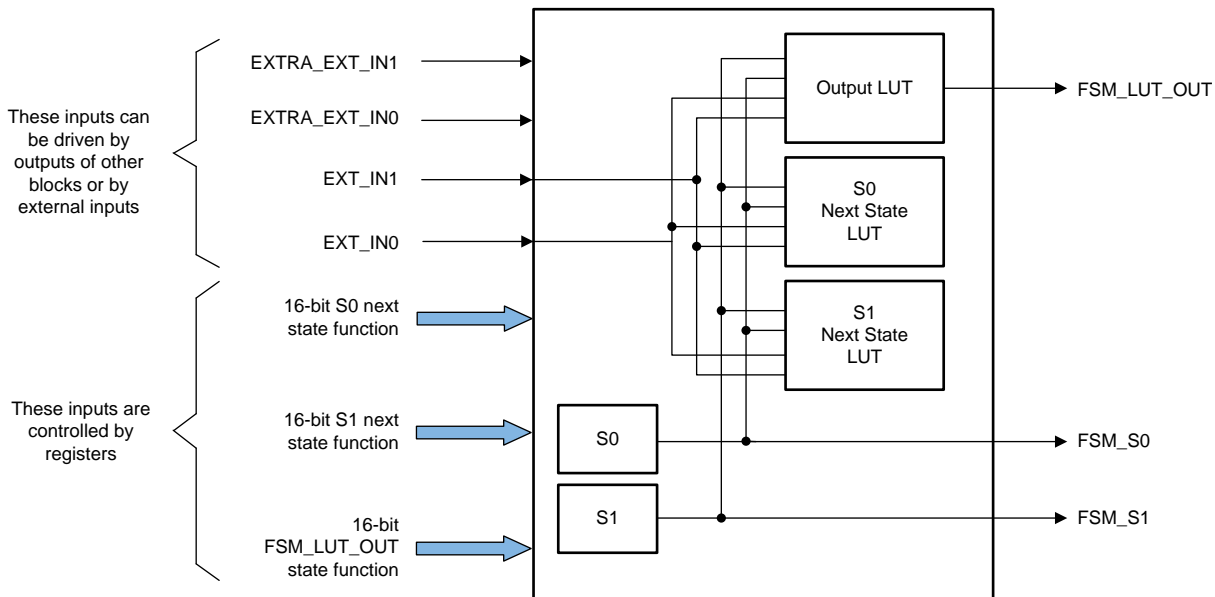
**Table 29-7. Counter Block Operating Modes**

EVENT	MODE_0	MODE_1	COUNT_EVENT_CTR L_x	COUNT_ADD_SHIFT_ x	COUNT_DIR_x	Action on CNTVAL
0	0	0	X	X	X	None
0	0	1	X	X	X	None
0	1	0	X	X	X	Count down
0	1	1	X	X	X	Count up
1	X	X	0	X	X	Load
1	X	X	1	0	0	Shift right
1	X	X	1	0	1	Shift left
1	X	X	1	1	0	Subtract
1	X	X	1	1	1	Add

### 29.4.3 FSM Block

The Finite State Machine (FSM) block provides the ability to build programmable finite state machines with up to four states. It has two register bits and two external inputs, and can be programmed either as two 2-state machines or as a single 4-state machines. For additional flexibility, there are two auxiliary inputs (EXTRA\_EXT\_IN0 and EXTRA\_EXT\_IN1) that can be used to create larger combinational functions by giving up a state functionality. The structure of the FSM is shown in the figure below.

Figure 29-9. FSM Block



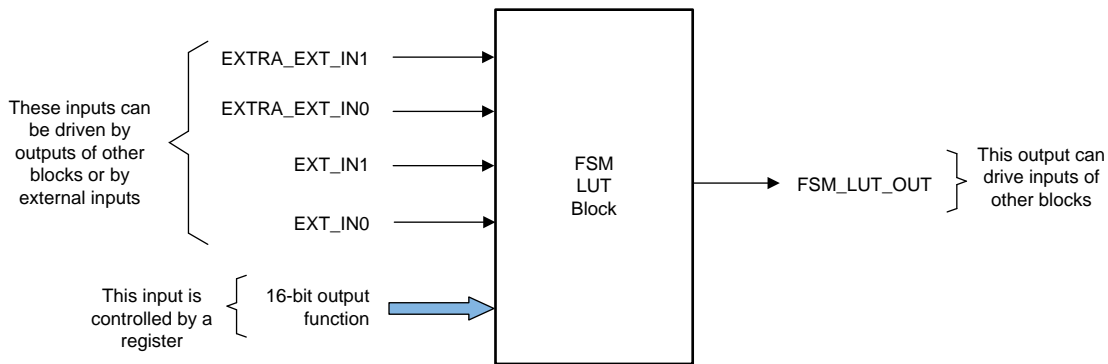
The signals and functionality of the FSM block are described below:

- **EXT\_IN0 and EXT\_IN1:** These are the two external inputs that can be used to control the output FSM\_LUT\_OUT or either of the states S0 and S1.
- **S0 and S1** are two state bits that have independent state control equations.
- **16-bit S0 equation** defines a function (EXT\_IN1, EXT\_IN0, S1, S0). The four bits in the order defined are used as an index into the 16-bit register to decide the next state of S0.
- **16-bit S1 equation** defines a function (EXT\_IN1, EXT\_IN0, S1, S0). The four bits in the order defined are used as an index into the 16-bit register to decide the next state of S1.
- **16-bit output equation** defines a function (EXT\_IN1, EXT\_IN0, S1, S0). The four bits in the order defined are used as an index into the 16-bit register to decide the output value of FSM\_LUT\_OUT. An additional level of configurability is provided such that FSM\_LUT\_OUT can use extra inputs in case the states S0 and S1 are unused.

One extra bit is used to select EXTRA\_EXT\_IN0 instead of S0. One extra bit is used to select EXTRA\_EXT\_IN1 instead of S1. Using these, one can effectively build 3-input or a 4-input LUT for the FSM\_LUT\_OUT by giving up one or two state bits, respectively.

The CFG\_MISC\_CTRL register controls the operation of the FSM block. Two bits in this register are used for each FSM Block to determine whether the FSM output LUT function uses the state variable S0/S1, or the corresponding extra external input signal FSM\_EXTRA\_EXT\_INx. '0' means use the state bit, and '1' means use the FSM\_EXTRA\_EXT\_IN0 / FSM\_EXTRA\_EXT\_IN1 signal.

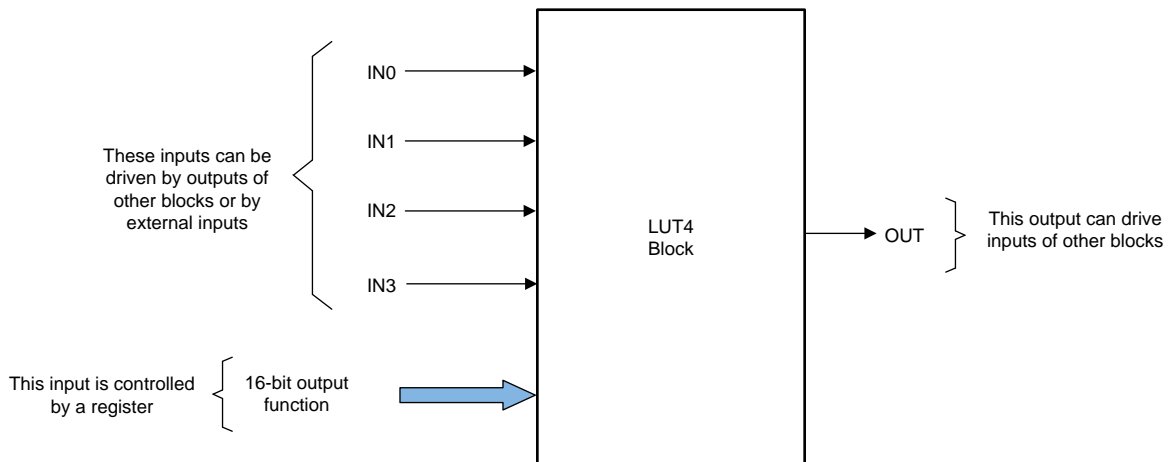
Figure 29-10. FSM LUT Block



#### 29.4.4 LUT4 Block

This is a simple four input Look-Up table (LUT) block with inputs IN0, IN1, IN2, and IN3. Any combinatorial Boolean equation using the four inputs can be realized by programming the 16-bit control register associated with each LUT4 block. For more information, see the LUT4 register descriptions located in [Section 29.6](#).

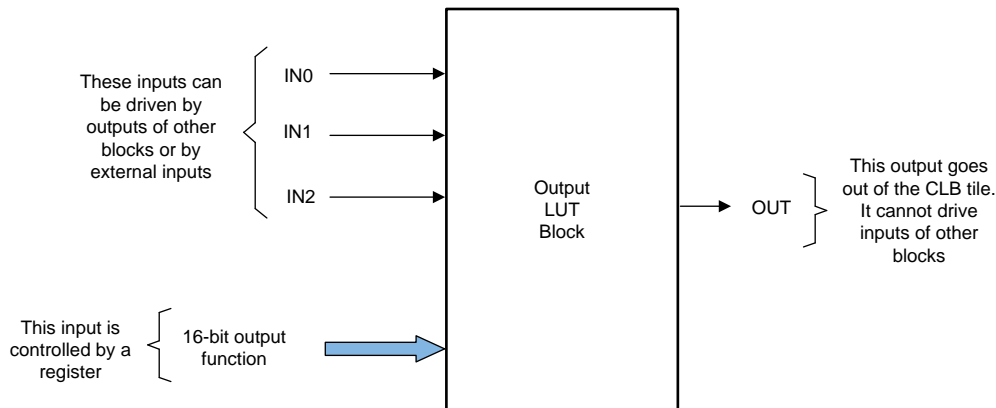
Figure 29-11. LUT4 Block



#### 29.4.5 Output LUT Block

The output LUT block is very similar in functionality to the LUT4 block, except that it has three inputs. Unlike the other sub blocks, the outputs of these blocks are meant to go out of the tile and hence they cannot be used by any other block within the tile. Any combinatorial function of the three inputs can be realized by the output LUT block. For more information, see the output LUT register descriptions located in [Section 29.6](#).

Figure 29-12. Output LUT Block



### 29.4.6 High Level Controller (HLC)

The high level Controller (HLC) is significantly more complex than the other blocks in the CLB tile. It performs two main functions:

- Provides a means of communication and data exchange with the rest of the device.
- Provides a programmable, event-based action system, which performs computation, manipulation of logic functionality, and data movement. In other words, events can be configured to trigger a predefined set of actions in the CLB tile, or to initiate data exchange with the rest of the device.

The architecture of the HLC is shown in Figure 29-14. It is an event-based system capable of handling up to four simultaneous events that can be selected from any outputs of the other blocks within the tile or from an external input.

Figure 29-13. High Level Controller Block

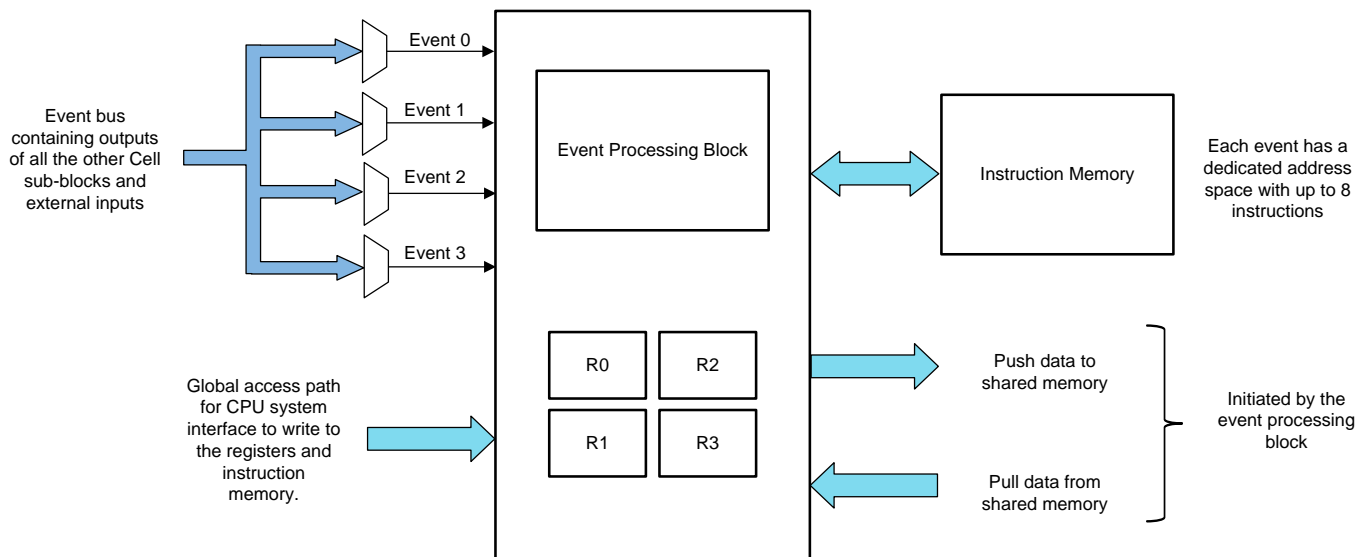


Figure 29-14.

#### 29.4.6.1 High Level Controller Events

Each of the four HLC events has a dedicated address from which instructions are executed. Event 0 is the highest priority and Event 3 is the lowest priority. Events are selected from the set of signals shown in Table 29-8. The lowest numbered event (Event 0) has the highest priority, and the highest numbered event (Event 3) has the lowest priority.

**Table 29-8. HLC Event List**

Index	HLC Event Mux
0	Always '0'
1	COUNTER_0 MATCH2
2	COUNTER_0 ZERO
3	COUNTER_0 MATCH1
4	FSM_0 STATE_BIT_0
5	FSM_0 STATE_BIT_1
6	FSM_0 LUT output
7	LUT4_0 output
8	Always '1'
9	COUNTER_1 MATCH2
10	COUNTER_1 ZERO
11	COUNTER_1 MATCH1
12	FSM_1 STATE_BIT_0
13	FSM_1 STATE_BIT_1
14	FSM_1 LUT output
15	LUT4_1 output
16	Always '0'
17	COUNTER_2 MATCH2
18	COUNTER_2 ZERO
19	COUNTER_2 MATCH1
20	FSM_2 STATE_BIT_0
21	FSM_2 STATE_BIT_1
22	FSM_2 LUT output
23	LUT4_2 output
24	External Input 0
25	External Input 1
26	External Input 2
27	External Input 3
28	External Input 4
29	External Input 5
30	External Input 6
31	External Input 7

### 29.4.6.2 High Level Controller Instructions

The instruction memory supports up to eight instructions per event. Each instruction sequence gets triggered on the rising edge of the corresponding event.

The HLC memory supports up to eight instructions per event, starting at the beginning of the fixed address range shown in [Table 29-9](#). An instruction sequence is triggered on the rising edge of the corresponding event. If two or more events occur simultaneously, the associated instruction sequences will each be executed sequentially in priority order.

**Table 29-9. HLC Instruction Address Ranges**

Address	Instructions for
00000 to 00111	Event 0
01000 to 01111	Event 1
10000 to 10111	Event 2
11000 to 11111	Event 3

The instruction format is shown in [Table 29-10](#).

**Table 29-10. Instruction Format**

Last Instruction Bit	5-Bit Opcode	3-Bit Source	3-Bit Destination
This bit set to 1 will stop execution after the current instruction	MOV 00000 T1_MOV 00001 T2_MOV 00010 PUSH 00011 PULL 00100 ADD 00101 SUB 00110 INTR 00111	Source can be R0, R1, R2, R3, C0, C1, C2	Destination can be R0, R1, R2, R3, C0, C1, C2 Note that for ADD/SUB instructions, only R0 to R3 can be the destination.

R0, R1, R2, and R3 are four 32-bit general purpose registers in the HLC. C0, C1, and C2 are three counter registers present in the CLB tile. <Src> is used to indicate the source and <Dest> is used to indicate the destination.

- **MOV <Src> <Dest>**  
This instruction moves <Src> to <Dest>. Both <Src> and <Dest> can be any of R0, R1, R2, R3, C0, C1, or C2.
- **MOV\_T1 <Src> <Dest>**  
This instruction moves <Src> to the Match1 register of the <Dest> counter. <Src> can be any of the registers R0, R1, R2, R3, or the counter values associated with C0, C1, or C2. <Dest> is the Match1 register of any of the counters C0, C1, or C2. Examples are:
  - This instruction moves the count value in C1 into register R0.  
MOV\_T1 C1 R0
  - This instruction moves the value in R2 into the Match1 register of counter C0.  
MOV\_T1 R2 C0
- **MOV\_T2 <Src> <Dest>**  
This instruction is similar to “MOV\_T1”, above. It moves <Src> to the Match2 register of the <Dest> counter. <Src> can be any of the registers R0, R1, R2, R3, or the counter values associated with C0, C1, or C2. <Dest> is the Match2 register of any of the counters C0, C1, or C2.
- **ADD <Src> <Dest>**  
This instruction performs an unsigned 32-bit addition. <Dest> = <Dest> + <Src>. Both <Src> and <Dest> can be R0, R1, R2, R3, C0, C1, or C2.
- **SUB <Src> <Dest>**  
This instruction performs an unsigned 32-bit subtraction. <Dest> = <Dest> - <Src> Both <Src> and <Dest> can be R0, R1, R2, R3, C0, C1, or C2.
- **PUSH <Src>**  
This instruction transfers data from <Src> to the data exchange push memory buffer in the CPU interface. <Src> can be any of R0, R1, R2, R3, C0, C1, or C2. The PUSH instruction is used as seen from the High Level Controller and pushes data into an internal 4 word FIFO.
- **PULL <Dest>**  
This instruction transfers data from to the data exchange pull memory buffer in the CPU interface to the <Dest> register. <Dest> can be any of R0, R1, R2, R3, C0, C1, or C2. The PULL instruction is used as seen from the High Level Controller and a PULL operation reads (pulls) data from an internal 4-word FIFO.
- **INTR <6-bit constant>**  
This instruction will flag an interrupt through the CPU interface. The 6-bit constant is stored in the interrupt flag register CLB\_INTR\_TAG\_REG.

### 29.4.6.3 <Src> and <Dest>

Three bits are used to encode the <Src> and <Dest> registers as shown in [Section 29.6](#).

**Table 29-11. HLC Register Encoding**

Register	Bits
R0	000
R1	001
R2	010
R3	011
C0	100
C1	101
C2	110

#### 29.4.6.4 Operation of the PUSH and PULL Instructions (overflow and underflow detection)

The PUSH and PULL operations of the HLC are intended for data exchange with the host system. There are separate FIFO buffers for PUSH and PULL operations. For example, a series of PUSH operations will write to successive locations in a linearly mapped memory buffer. The PUSH buffer is mapped at address offset 0x200 and the PULL buffer is mapped at address offset 0x300.

The CPU can read from and write to the PUSH and PULL buffers respectively, in order to exchange data with the HLC. Data pushed by the HLC is read by the CPU from the PUSH buffers. Data sent from the CPU to the HLC will be written by the CPU to the PULL buffer and will be read by the HLC using the PULL instruction.

There are separate PUSH and PULL address pointers that increment each time the HLC performs a PUSH or PULL operation. These address pointers are also memory-mapped so that the CPU can determine their value. They are also writable and can be reset by the CPU at any time.

Overflow and underflow detection is done by simply reading the values of the PUSH and PULL address pointers.

In the CLB module of the device, the depth of the PUSH and PULL FIFOs is four 32-bit words each. If the CPU starts a fresh data transfer to the PULL buffers and sees the address pointer greater than four, then an underflow has occurred since the HLC has pulled more data than the number of words written by the CPU into the buffer.

## 29.5 CPU Interface

### 29.5.1 Register Description

There are three classes of registers that are used to control and configure the CLB tile. This specification only describes the offset addresses of the registers. The absolute register addresses will be different for each CLB tile. The three instances of the various blocks (LUT4, FSM, and Counter Block) are numbered 0, 1, and 2.

- Logic configuration registers (0x000 – 0x0FF)

These registers control the core reconfiguration logic for the tile. All registers in this group are EALLOW protected and also protected by the LOCK register.

- Top level control registers (0x100 – 0x1FF)

These registers are used for top level and device related control of the CLB. These registers typically control mux selects for inputs, global enables, and so forth, and are accessible by normal memory mapped access. Some of these registers have EALLOW and LOCK protection.

- Data exchange registers (0x200 – 0x3FF)

These registers are used to exchange data between the CLB and the rest of the device. They are accessible by normal memory mapped access and no EALLOW or LOCK protection exists.

**NOTE:** EALLOW protection means that the write access to the register will be enabled only when the EALLOW instruction has been executed prior to the write access. The complementary EDIS instruction disables access to all registers protected in this way. For more information, see [Section 29.6](#).

### 29.5.2 Non-Memory Mapped Registers

The memory mapped CLB registers are described later in this document, however many of the CLB resources including counters, the instruction memory of the High Level controller, and the HLC general purpose registers (R0 through R3) are only indirectly accessible through a local interface bus and are not memory mapped. These registers are accessible through the two memory mapped registers CLB\_LOAD\_DATA and CLB\_LOAD\_ADDR.

The user loads the data to be written into the CLB\_LOAD\_DATA register, then loads the appropriate address into CLB\_LOAD\_ADDR to determine where this data is written. Writing a '1' to bit position 0 in the CLB\_LOAD\_EN register then causes an internal write operation to be triggered. The address allocation for the CLB\_LOAD\_ADDR register is shown in [Table 29-12](#).

**Table 29-12. Non-Memory Mapped Register Addresses**

Address (Binary)	Resource
000000 to 000010	Counter 0 to 2 load value
000100 to 000110	Counter 0 to 2 Match1 value
001000 to 001010	Counter 0 to 2 Match2 value
001100 to 001111	R0 to R3 of High Level controller
100000 to 100111	Instructions for Event 0
101000 to 101111	Instructions for Event 1
110000 to 110111	Instructions for Event 2
111000 to 111111	Instructions for Event 3

Use the following steps to load the value 0x11223344 into the general purpose R0 register:

1. Write 0x11223344 to CLB\_LOAD\_DATA.
2. Write 0xc to CLB\_LOAD\_ADDR.
3. Write 0x1 to CLB\_LOAD\_EN.



## 29.6 CLB Registers

This section describes the Configurable Logic Block Registers.

### 29.6.1 CLB Base Addresses

This is the Configurable Logic Block Base Address Table.

**Table 29-13. CLB Base Address Table (C28)**

Bit Field Name		DriverLib Name	Base Address
Instance	Structure		
Clb1LogicCfgRegs	CLB_LOGIC_CONFIG_REGS	CLB1_LOGICCFG_BASE	0x0000_3000
Clb1LogicCtrlRegs	CLB_LOGIC_CONTROL_REGS	CLB1_LOGICCTRL_BASE	0x0000_3100
Clb1DataExchRegs	CLB_DATA_EXCHANGE_REGS	CLB1_DATAEXCH_BASE	0x0000_3200
Clb2LogicCfgRegs	CLB_LOGIC_CONFIG_REGS	CLB2_LOGICCFG_BASE	0x0000_3400
Clb2LogicCtrlRegs	CLB_LOGIC_CONTROL_REGS	CLB2_LOGICCTRL_BASE	0x0000_3500
Clb2DataExchRegs	CLB_DATA_EXCHANGE_REGS	CLB2_DATAEXCH_BASE	0x0000_3600
Clb3LogicCfgRegs	CLB_LOGIC_CONFIG_REGS	CLB3_LOGICCFG_BASE	0x0000_3800
Clb3LogicCtrlRegs	CLB_LOGIC_CONTROL_REGS	CLB3_LOGICCTRL_BASE	0x0000_3900
Clb3DataExchRegs	CLB_DATA_EXCHANGE_REGS	CLB3_DATAEXCH_BASE	0x0000_3A00
Clb4LogicCfgRegs	CLB_LOGIC_CONFIG_REGS	CLB4_LOGICCFG_BASE	0x0000_3C00
Clb4LogicCtrlRegs	CLB_LOGIC_CONTROL_REGS	CLB4_LOGICCTRL_BASE	0x0000_3D00
Clb4DataExchRegs	CLB_DATA_EXCHANGE_REGS	CLB4_DATAEXCH_BASE	0x0000_3E00

## 29.6.2 CLB\_LOGIC\_CONFIG\_REGS Registers

Table 29-14 lists the CLB\_LOGIC\_CONFIG\_REGS registers. All register offset addresses not listed in Table 29-14 should be considered as reserved locations and the register contents should not be modified.

**Table 29-14. CLB\_LOGIC\_CONFIG\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
2h	CLB_COUNT_RESET	Counter Block RESET	EALLOW, LOCK	<a href="#">Go</a>
4h	CLB_COUNT_MODE_1	Counter Block MODE_1	EALLOW, LOCK	<a href="#">Go</a>
6h	CLB_COUNT_MODE_0	Counter Block MODE_0	EALLOW, LOCK	<a href="#">Go</a>
8h	CLB_COUNT_EVENT	Counter Block EVENT	EALLOW, LOCK	<a href="#">Go</a>
Ah	CLB_FSM_EXTRA_IN0	FSM Extra EXT_IN0	EALLOW, LOCK	<a href="#">Go</a>
Ch	CLB_FSM_EXTERNAL_IN0	FSM EXT_IN0	EALLOW, LOCK	<a href="#">Go</a>
Eh	CLB_FSM_EXTERNAL_IN1	FSM_EXT_IN1	EALLOW, LOCK	<a href="#">Go</a>
10h	CLB_FSM_EXTRA_IN1	FSM Extra_EXT_IN1	EALLOW, LOCK	<a href="#">Go</a>
12h	CLB_LUT4_IN0	LUT4_0/1/2 IN0 input source	EALLOW, LOCK	<a href="#">Go</a>
14h	CLB_LUT4_IN1	LUT4_0/1/2 IN1 input source	EALLOW, LOCK	<a href="#">Go</a>
16h	CLB_LUT4_IN2	LUT4_0/1/2 IN2 input source	EALLOW, LOCK	<a href="#">Go</a>
18h	CLB_LUT4_IN3	LUT4_0/1/2 IN3 input source	EALLOW, LOCK	<a href="#">Go</a>
1Ch	CLB_FSM_LUT_FN1_0	LUT function for FSM Unit 1 and Unit 0	EALLOW, LOCK	<a href="#">Go</a>
1Eh	CLB_FSM_LUT_FN2	LUT function for FSM Unit 2	EALLOW, LOCK	<a href="#">Go</a>
20h	CLB_LUT4_FN1_0	LUT function for LUT4 block of Unit 1 and 0	EALLOW, LOCK	<a href="#">Go</a>
22h	CLB_LUT4_FN2	LUT function for LUT4 block of Unit 2	EALLOW, LOCK	<a href="#">Go</a>
24h	CLB_FSM_NEXT_STATE_0	FSM Next state equations for Unit 0	EALLOW, LOCK	<a href="#">Go</a>
26h	CLB_FSM_NEXT_STATE_1	FSM Next state equations for Unit 1	EALLOW, LOCK	<a href="#">Go</a>
28h	CLB_FSM_NEXT_STATE_2	FSM Next state equations for Unit 2	EALLOW, LOCK	<a href="#">Go</a>
2Ah	CLB_MISC_CONTROL	Static controls for Ctr,FSM	EALLOW, LOCK	<a href="#">Go</a>
2Ch	CLB_OUTPUT_LUT_0	Inp Sel, LUT fns for Out0	EALLOW, LOCK	<a href="#">Go</a>
2Eh	CLB_OUTPUT_LUT_1	Inp Sel, LUT fns for Out1	EALLOW, LOCK	<a href="#">Go</a>
30h	CLB_OUTPUT_LUT_2	Inp Sel, LUT fns for Out2	EALLOW, LOCK	<a href="#">Go</a>
32h	CLB_OUTPUT_LUT_3	Inp Sel, LUT fns for Out3	EALLOW, LOCK	<a href="#">Go</a>
34h	CLB_OUTPUT_LUT_4	Inp Sel, LUT fns for Out4	EALLOW, LOCK	<a href="#">Go</a>
36h	CLB_OUTPUT_LUT_5	Inp Sel, LUT fns for Out5	EALLOW, LOCK	<a href="#">Go</a>
38h	CLB_OUTPUT_LUT_6	Inp Sel, LUT fns for Out6	EALLOW, LOCK	<a href="#">Go</a>
3Ah	CLB_OUTPUT_LUT_7	Inp Sel, LUT fns for Out7	EALLOW, LOCK	<a href="#">Go</a>
3Ch	CLB_HLC_EVENT_SEL	Event Selector register for the High Level controller	EALLOW, LOCK	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 29-15 shows the codes that are used for access types in this section.

**Table 29-15. CLB\_LOGIC\_CONFIG\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
<b>Write Type</b>		
W	W	Write
W1C	W1C	Write 1 to clear

**Table 29-15. CLB\_LOGIC\_CONFIG\_REGS Access Type Codes (continued)**

Access Type	Code	Description
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**29.6.2.1 CLB\_COUNT\_RESET Register (Offset = 2h) [reset = 0h]**

CLB\_COUNT\_RESET is shown in [Figure 29-15](#) and described in [Table 29-16](#).

Return to the [Summary Table](#).

Counter Block RESET

**Figure 29-15. CLB\_COUNT\_RESET 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																	SEL_2			SEL_1			SEL_0								
R/W1C-0h																	R/W-0h			R/W-0h			R/W-0h								

**Table 29-16. CLB\_COUNT\_RESET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	Counter reset select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	Counter reset select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	Counter reset select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.2 CLB\_COUNT\_MODE\_1 Register (Offset = 4h) [reset = 0h]**

CLB\_COUNT\_MODE\_1 is shown in [Figure 29-16](#) and described in [Table 29-17](#).

Return to the [Summary Table](#).

Counter Block MODE\_1

**Figure 29-16. CLB\_COUNT\_MODE\_1 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															SEL_2				SEL_1				SEL_0								
R/W1C-0h															R/W-0h				R/W-0h				R/W-0h								

**Table 29-17. CLB\_COUNT\_MODE\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	Counter MODE_1 select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	Counter MODE_1 select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	Counter MODE_1 select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

### 29.6.2.3 CLB\_COUNT\_MODE\_0 Register (Offset = 6h) [reset = 0h]

CLB\_COUNT\_MODE\_0 is shown in [Figure 29-17](#) and described in [Table 29-18](#).

Return to the [Summary Table](#).

Counter Block MODE\_0

**Figure 29-17. CLB\_COUNT\_MODE\_0 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																	SEL_2			SEL_1			SEL_0								
R/W1C-0h																	R/W-0h			R/W-0h			R/W-0h								

**Table 29-18. CLB\_COUNT\_MODE\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	Counter MODE_0 select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	Counter MODE_0 select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	Counter MODE_0 select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.4 CLB\_COUNT\_EVENT Register (Offset = 8h) [reset = 0h]**

CLB\_COUNT\_EVENT is shown in [Figure 29-18](#) and described in [Table 29-19](#).

Return to the [Summary Table](#).

Counter Block EVENT

**Figure 29-18. CLB\_COUNT\_EVENT 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															SEL_2			SEL_1			SEL_0										
R/W1C-0h															R/W-0h			R/W-0h			R/W-0h										

**Table 29-19. CLB\_COUNT\_EVENT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	Counter event select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	Counter event select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	Counter event select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

### 29.6.2.5 CLB\_FSM\_EXTRA\_IN0 Register (Offset = Ah) [reset = 0h]

CLB\_FSM\_EXTRA\_IN0 is shown in [Figure 29-19](#) and described in [Table 29-20](#).

Return to the [Summary Table](#).

FSM Extra EXT\_IN0

**Figure 29-19. CLB\_FSM\_EXTRA\_IN0 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																	SEL_2			SEL_1			SEL_0								
R/W1C-0h																	R/W-0h			R/W-0h			R/W-0h								

**Table 29-20. CLB\_FSM\_EXTRA\_IN0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	FSM block extra external IN0 select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	FSM block extra external IN0 select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	FSM block extra external IN0 select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn



**29.6.2.6 CLB\_FSM\_EXTERNAL\_IN0 Register (Offset = Ch) [reset = 0h]**

CLB\_FSM\_EXTERNAL\_IN0 is shown in [Figure 29-20](#) and described in [Table 29-21](#).

Return to the [Summary Table](#).

FSM EXT\_IN0

**Figure 29-20. CLB\_FSM\_EXTERNAL\_IN0 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																	SEL_2			SEL_1			SEL_0								
R/W1C-0h																	R/W-0h			R/W-0h			R/W-0h								

**Table 29-21. CLB\_FSM\_EXTERNAL\_IN0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	FSM block EXT_IN0 select input for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	FSM block EXT_IN0 select input for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	FSM block EXT_IN0 select input for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.7 CLB\_FSM\_EXTERNAL\_IN1 Register (Offset = Eh) [reset = 0h]**

CLB\_FSM\_EXTERNAL\_IN1 is shown in [Figure 29-21](#) and described in [Table 29-22](#).

Return to the [Summary Table](#).

FSM\_EXT\_IN1

**Figure 29-21. CLB\_FSM\_EXTERNAL\_IN1 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																	SEL_2			SEL_1			SEL_0								
R/W1C-0h																	R/W-0h			R/W-0h			R/W-0h								

**Table 29-22. CLB\_FSM\_EXTERNAL\_IN1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	FSM block EXT_IN1 select input for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	FSM block EXT_IN1 select input for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	FSM block EXT_IN1 select input for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.8 CLB\_FSM\_EXTRA\_IN1 Register (Offset = 10h) [reset = 0h]**

CLB\_FSM\_EXTRA\_IN1 is shown in [Figure 29-22](#) and described in [Table 29-23](#).

Return to the [Summary Table](#).

FSM Extra\_EXT\_IN1

**Figure 29-22. CLB\_FSM\_EXTRA\_IN1 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															SEL_2				SEL_1				SEL_0								
R/W1C-0h															R/W-0h				R/W-0h				R/W-0h								

**Table 29-23. CLB\_FSM\_EXTRA\_IN1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	FSM block extra external IN1 select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	FSM block extra external IN1 select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	FSM block extra external IN1 select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.9 CLB\_LUT4\_IN0 Register (Offset = 12h) [reset = 0h]**

CLB\_LUT4\_IN0 is shown in [Figure 29-23](#) and described in [Table 29-24](#).

Return to the [Summary Table](#).

LUT4\_0/1/2 IN0 input source

**Figure 29-23. CLB\_LUT4\_IN0 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																	SEL_2			SEL_1			SEL_0								
R/W1C-0h																	R/W-0h			R/W-0h			R/W-0h								

**Table 29-24. CLB\_LUT4\_IN0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	LUT4 block IN0 select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	LUT4 block IN0 select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	LUT4 block IN0 select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.10 CLB\_LUT4\_IN1 Register (Offset = 14h) [reset = 0h]**

CLB\_LUT4\_IN1 is shown in [Figure 29-24](#) and described in [Table 29-25](#).

Return to the [Summary Table](#).

LUT4\_0/1/2 IN1 input source

**Figure 29-24. CLB\_LUT4\_IN1 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															SEL_2			SEL_1			SEL_0										
R/W1C-0h															R/W-0h			R/W-0h			R/W-0h										

**Table 29-25. CLB\_LUT4\_IN1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	LUT4 block IN1 select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	LUT4 block IN1 select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	LUT4 block IN1 select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.11 CLB\_LUT4\_IN2 Register (Offset = 16h) [reset = 0h]**

CLB\_LUT4\_IN2 is shown in [Figure 29-25](#) and described in [Table 29-26](#).

Return to the [Summary Table](#).

LUT4\_0/1/2 IN2 input source

**Figure 29-25. CLB\_LUT4\_IN2 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																	SEL_2			SEL_1			SEL_0								
R/W1C-0h																	R/W-0h			R/W-0h			R/W-0h								

**Table 29-26. CLB\_LUT4\_IN2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	LUT4 block IN2 select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	LUT4 block IN2 select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	LUT4 block IN2 select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

### 29.6.2.12 CLB\_LUT4\_IN3 Register (Offset = 18h) [reset = 0h]

CLB\_LUT4\_IN3 is shown in [Figure 29-26](#) and described in [Table 29-27](#).

Return to the [Summary Table](#).

LUT4\_0/1/2 IN3 input source

**Figure 29-26. CLB\_LUT4\_IN3 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															SEL_2				SEL_1				SEL_0								
R/W1C-0h															R/W-0h				R/W-0h				R/W-0h								

**Table 29-27. CLB\_LUT4\_IN3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R/W1C	0h	Reserved
14-10	SEL_2	R/W	0h	LUT4 block IN3 select inputs for unit 2. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	SEL_1	R/W	0h	LUT4 block IN3 select inputs for unit 1. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	SEL_0	R/W	0h	LUT4 block IN3 select inputs for unit 0. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.13 CLB\_FSM\_LUT\_FN1\_0 Register (Offset = 1Ch) [reset = 0h]**

CLB\_FSM\_LUT\_FN1\_0 is shown in [Figure 29-27](#) and described in [Table 29-28](#).

Return to the [Summary Table](#).

LUT function for FSM Unit 1 and Unit 0

**Figure 29-27. CLB\_FSM\_LUT\_FN1\_0 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
FN1																FN0															
R/W-0h																R/W-0h															

**Table 29-28. CLB\_FSM\_LUT\_FN1\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	FN1	R/W	0h	FSM block LUT output function for unit 1 Reset type: SYSRSn
15-0	FN0	R/W	0h	FSM block LUT output function for unit 0 Reset type: SYSRSn



**29.6.2.14 CLB\_FSM\_LUT\_FN2 Register (Offset = 1Eh) [reset = 0h]**

CLB\_FSM\_LUT\_FN2 is shown in [Figure 29-28](#) and described in [Table 29-29](#).

Return to the [Summary Table](#).

LUT function for FSM Unit 2

**Figure 29-28. CLB\_FSM\_LUT\_FN2 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																FN1															
R/W1C-0h																R/W-0h															

**Table 29-29. CLB\_FSM\_LUT\_FN2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W1C	0h	Reserved
15-0	FN1	R/W	0h	LUT4 output function for unit 2 Reset type: SYSRSn

**29.6.2.15 CLB\_LUT4\_FN1\_0 Register (Offset = 20h) [reset = 0h]**

CLB\_LUT4\_FN1\_0 is shown in [Figure 29-29](#) and described in [Table 29-30](#).

Return to the [Summary Table](#).

LUT function for LUT4 block of Unit 1 and 0

**Figure 29-29. CLB\_LUT4\_FN1\_0 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
FN1																FN0															
R/W-0h																R/W-0h															

**Table 29-30. CLB\_LUT4\_FN1\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	FN1	R/W	0h	LUT4 output function for unit 1 Reset type: SYSRSn
15-0	FN0	R/W	0h	LUT4 output function for unit 0 Reset type: SYSRSn

**29.6.2.16 CLB\_LUT4\_FN2 Register (Offset = 22h) [reset = 0h]**

CLB\_LUT4\_FN2 is shown in [Figure 29-30](#) and described in [Table 29-31](#).

Return to the [Summary Table](#).

LUT function for LUT4 block of Unit 2

**Figure 29-30. CLB\_LUT4\_FN2 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																FN1															
R/W1C-0h																R/W-0h															

**Table 29-31. CLB\_LUT4\_FN2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W1C	0h	Reserved
15-0	FN1	R/W	0h	LUT4 output function for unit 2 Reset type: SYSRSn

**29.6.2.17 CLB\_FSM\_NEXT\_STATE\_0 Register (Offset = 24h) [reset = 0h]**

CLB\_FSM\_NEXT\_STATE\_0 is shown in [Figure 29-31](#) and described in [Table 29-32](#).

Return to the [Summary Table](#).

FSM Next state equations for Unit 0

**Figure 29-31. CLB\_FSM\_NEXT\_STATE\_0 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
S1																S0															
R/W-0h																R/W-0h															

**Table 29-32. CLB\_FSM\_NEXT\_STATE\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	S1	R/W	0h	FSM next state function for S1, unit0 Reset type: SYSRSn
15-0	S0	R/W	0h	FSM next state function for S0, unit0 Reset type: SYSRSn

**29.6.2.18 CLB\_FSM\_NEXT\_STATE\_1 Register (Offset = 26h) [reset = 0h]**

CLB\_FSM\_NEXT\_STATE\_1 is shown in [Figure 29-32](#) and described in [Table 29-33](#).

Return to the [Summary Table](#).

FSM Next state equations for Unit 1

**Figure 29-32. CLB\_FSM\_NEXT\_STATE\_1 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
S1																S0															
R/W-0h																R/W-0h															

**Table 29-33. CLB\_FSM\_NEXT\_STATE\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	S1	R/W	0h	FSM next state function for S1, unit1 Reset type: SYSRSn
15-0	S0	R/W	0h	FSM next state function for S0, unit1 Reset type: SYSRSn

**29.6.2.19 CLB\_FSM\_NEXT\_STATE\_2 Register (Offset = 28h) [reset = 0h]**

CLB\_FSM\_NEXT\_STATE\_2 is shown in [Figure 29-33](#) and described in [Table 29-34](#).

Return to the [Summary Table](#).

FSM Next state equations for Unit 2

**Figure 29-33. CLB\_FSM\_NEXT\_STATE\_2 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
S1																S0															
R/W-0h																R/W-0h															

**Table 29-34. CLB\_FSM\_NEXT\_STATE\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	S1	R/W	0h	FSM next state function for S1, unit2 Reset type: SYSRSn
15-0	S0	R/W	0h	FSM next state function for S0, unit2 Reset type: SYSRSn

### 29.6.2.20 CLB\_MISC\_CONTROL Register (Offset = 2Ah) [reset = 0h]

CLB\_MISC\_CONTROL is shown in [Figure 29-34](#) and described in [Table 29-35](#).

Return to the [Summary Table](#).

Static controls for Ctr,FSM

**Figure 29-34. CLB\_MISC\_CONTROL Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED						FSM_EXTRA_SEL1_2	FSM_EXTRA_SELO_2
R-0-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
FSM_EXTRA_SEL1_1	FSM_EXTRA_SELO_1	FSM_EXTRA_SEL1_0	FSM_EXTRA_SELO_0	COUNT_SERIALIZER_2	COUNT_SERIALIZER_1	COUNT_SERIALIZER_0	COUNT_EVENT_CTRL_2
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
COUNT_DIR_2	COUNT_ADD_SHIFT_2	COUNT_EVENT_CTRL_1	COUNT_DIR_1	COUNT_ADD_SHIFT_1	COUNT_EVENT_CTRL_0	COUNT_DIR_0	COUNT_ADD_SHIFT_0
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 29-35. CLB\_MISC\_CONTROL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-18	RESERVED	R-0	0h	Reserved
17	FSM_EXTRA_SEL1_2	R/W	0h	Defines which input should be selected for the FSM LUT of UNIT 2 0 = Selects State S1 for the FSM LUT 1 = Selects EXTRA_EXT_IN1 for the FSM LUT Reset type: SYSRSn
16	FSM_EXTRA_SELO_2	R/W	0h	Defines which input should be selected for the FSM LUT of UNIT 2 0 = Selects State S0 for the FSM LUT 1 = Selects EXTRA_EXT_IN0 for the FSM LUT Reset type: SYSRSn
15	FSM_EXTRA_SEL1_1	R/W	0h	Defines which input should be selected for the FSM LUT of UNIT 1 0 = Selects State S1 for the FSM LUT 1 = Selects EXTRA_EXT_IN1 for the FSM LUT Reset type: SYSRSn
14	FSM_EXTRA_SELO_1	R/W	0h	Defines which input should be selected for the FSM LUT of UNIT 1 0 = Selects State S0 for the FSM LUT 1 = Selects EXTRA_EXT_IN0 for the FSM LUT Reset type: SYSRSn
13	FSM_EXTRA_SEL1_0	R/W	0h	Defines which input should be selected for the FSM LUT of UNIT 0 0 = Selects State S1 for the FSM LUT 1 = Selects EXTRA_EXT_IN1 for the FSM LUT Reset type: SYSRSn
12	FSM_EXTRA_SELO_0	R/W	0h	Defines which input should be selected for the FSM LUT of UNIT 0 0 = Selects State S0 for the FSM LUT 1 = Selects EXTRA_EXT_IN0 for the FSM LUT Reset type: SYSRSn

**Table 29-35. CLB\_MISC\_CONTROL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	COUNT_SERIALIZER_2	R/W	0h	Controls if the Counter of UNIT 2 is the Serialzer mode or not. 0 = Normal mode 1 = Serializer mode Reset type: SYSRSn
10	COUNT_SERIALIZER_1	R/W	0h	Controls if the Counter of UNIT 1 is the Serialzer mode or not. 0 = Normal mode 1 = Serializer mode Reset type: SYSRSn
9	COUNT_SERIALIZER_0	R/W	0h	Controls if the Counter of UNIT 0 is the Serialzer mode or not. 0 = Normal mode 1 = Serializer mode Reset type: SYSRSn
8	COUNT_EVENT_CTRL_2	R/W	0h	Controls the actions on an EVENT for UNIT2. 0 = No add or shift, but load the predefined value 1 = Based on other bits, add/shift with the predefined value Reset type: SYSRSn
7	COUNT_DIR_2	R/W	0h	Controls add/shift direction for UNIT 2 0 = right shift or subtract 1 = left shift or add Reset type: SYSRSn
6	COUNT_ADD_SHIFT_2	R/W	0h	Controls whether the UNIT 2 counter will do an ADD or a SHIFT on an EVENT. 0 =Shift 1 = ADD Reset type: SYSRSn
5	COUNT_EVENT_CTRL_1	R/W	0h	Controls the actions on an EVENT for UNIT1. 0 = No add or shift, but load the predefined value 1 = Based on other bits, add/shift with the predefined value Reset type: SYSRSn
4	COUNT_DIR_1	R/W	0h	Controls add/shift direction for UNIT 1 0 = right shift or subtract 1 = left shift or add Reset type: SYSRSn
3	COUNT_ADD_SHIFT_1	R/W	0h	Controls whether the UNIT 1 counter will do an ADD or a SHIFT on an EVENT. 0 = Shift 1 = ADD Reset type: SYSRSn
2	COUNT_EVENT_CTRL_0	R/W	0h	Controls the actions on an EVENT for UNIT1. 0 = No add or shift, but load the predefined value 1 = Based on other bits, add/shift with the predefined value Reset type: SYSRSn
1	COUNT_DIR_0	R/W	0h	Controls add/shift direction for UNIT 0 0 = right shift or subtract 1 = left shift or add Reset type: SYSRSn



**Table 29-35. CLB\_MISC\_CONTROL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	COUNT_ADD_SHIFT_0	R/W	0h	Controls whether the UNIT 0 counter will do an ADD or a SHIFT on an EVENT. 0 = Shift 1 = ADD Reset type: SYSRSn

**29.6.2.21 CLB\_OUTPUT\_LUT\_0 Register (Offset = 2Ch) [reset = 0h]**

CLB\_OUTPUT\_LUT\_0 is shown in [Figure 29-35](#) and described in [Table 29-36](#).

Return to the [Summary Table](#).

Inp Sel, LUT fns for Out0

**Figure 29-35. CLB\_OUTPUT\_LUT\_0 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									FN						IN2				IN1				IN0								
R/W1C-0h									R/W-0h						R/W-0h				R/W-0h				R/W-0h								

**Table 29-36. CLB\_OUTPUT\_LUT\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R/W1C	0h	Reserved
22-15	FN	R/W	0h	Output function for output LUT Reset type: SYSRSn
14-10	IN2	R/W	0h	Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	IN1	R/W	0h	Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	IN0	R/W	0h	Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.22 CLB\_OUTPUT\_LUT\_1 Register (Offset = 2Eh) [reset = 0h]**

CLB\_OUTPUT\_LUT\_1 is shown in [Figure 29-36](#) and described in [Table 29-37](#).

Return to the [Summary Table](#).

Inp Sel, LUT fns for Out1

**Figure 29-36. CLB\_OUTPUT\_LUT\_1 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									FN						IN2				IN1				IN0								
R/W1C-0h									R/W-0h						R/W-0h				R/W-0h				R/W-0h								

**Table 29-37. CLB\_OUTPUT\_LUT\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R/W1C	0h	Reserved
22-15	FN	R/W	0h	Output function for output LUT Reset type: SYSRSn
14-10	IN2	R/W	0h	Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	IN1	R/W	0h	Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	IN0	R/W	0h	Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.23 CLB\_OUTPUT\_LUT\_2 Register (Offset = 30h) [reset = 0h]**

CLB\_OUTPUT\_LUT\_2 is shown in [Figure 29-37](#) and described in [Table 29-38](#).

Return to the [Summary Table](#).

Inp Sel, LUT fns for Out2

**Figure 29-37. CLB\_OUTPUT\_LUT\_2 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									FN						IN2				IN1				IN0								
R/W1C-0h									R/W-0h						R/W-0h				R/W-0h				R/W-0h								

**Table 29-38. CLB\_OUTPUT\_LUT\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R/W1C	0h	Reserved
22-15	FN	R/W	0h	Output function for output LUT Reset type: SYSRSn
14-10	IN2	R/W	0h	Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	IN1	R/W	0h	Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	IN0	R/W	0h	Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

### 29.6.2.24 CLB\_OUTPUT\_LUT\_3 Register (Offset = 32h) [reset = 0h]

CLB\_OUTPUT\_LUT\_3 is shown in [Figure 29-38](#) and described in [Table 29-39](#).

Return to the [Summary Table](#).

Inp Sel, LUT fns for Out3

**Figure 29-38. CLB\_OUTPUT\_LUT\_3 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									FN						IN2				IN1				IN0								
R/W1C-0h									R/W-0h						R/W-0h				R/W-0h				R/W-0h								

**Table 29-39. CLB\_OUTPUT\_LUT\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R/W1C	0h	Reserved
22-15	FN	R/W	0h	Output function for output LUT Reset type: SYSRSn
14-10	IN2	R/W	0h	Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	IN1	R/W	0h	Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	IN0	R/W	0h	Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.25 CLB\_OUTPUT\_LUT\_4 Register (Offset = 34h) [reset = 0h]**

CLB\_OUTPUT\_LUT\_4 is shown in [Figure 29-39](#) and described in [Table 29-40](#).

Return to the [Summary Table](#).

Inp Sel, LUT fns for Out4

**Figure 29-39. CLB\_OUTPUT\_LUT\_4 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									FN						IN2				IN1				IN0								
R/W1C-0h									R/W-0h						R/W-0h				R/W-0h				R/W-0h								

**Table 29-40. CLB\_OUTPUT\_LUT\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R/W1C	0h	Reserved
22-15	FN	R/W	0h	Output function for output LUT Reset type: SYSRSn
14-10	IN2	R/W	0h	Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	IN1	R/W	0h	Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	IN0	R/W	0h	Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.26 CLB\_OUTPUT\_LUT\_5 Register (Offset = 36h) [reset = 0h]**

CLB\_OUTPUT\_LUT\_5 is shown in [Figure 29-40](#) and described in [Table 29-41](#).

Return to the [Summary Table](#).

Inp Sel, LUT fns for Out5

**Figure 29-40. CLB\_OUTPUT\_LUT\_5 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									FN					IN2				IN1				IN0									
R/W1C-0h									R/W-0h					R/W-0h				R/W-0h				R/W-0h									

**Table 29-41. CLB\_OUTPUT\_LUT\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R/W1C	0h	Reserved
22-15	FN	R/W	0h	Output function for output LUT Reset type: SYSRSn
14-10	IN2	R/W	0h	Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	IN1	R/W	0h	Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	IN0	R/W	0h	Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.27 CLB\_OUTPUT\_LUT\_6 Register (Offset = 38h) [reset = 0h]**

CLB\_OUTPUT\_LUT\_6 is shown in [Figure 29-41](#) and described in [Table 29-42](#).

Return to the [Summary Table](#).

Inp Sel, LUT fns for Out6

**Figure 29-41. CLB\_OUTPUT\_LUT\_6 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									FN						IN2				IN1				IN0								
R/W1C-0h									R/W-0h						R/W-0h				R/W-0h				R/W-0h								

**Table 29-42. CLB\_OUTPUT\_LUT\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R/W1C	0h	Reserved
22-15	FN	R/W	0h	Output function for output LUT Reset type: SYSRSn
14-10	IN2	R/W	0h	Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	IN1	R/W	0h	Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	IN0	R/W	0h	Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn



**29.6.2.28 CLB\_OUTPUT\_LUT\_7 Register (Offset = 3Ah) [reset = 0h]**

CLB\_OUTPUT\_LUT\_7 is shown in [Figure 29-42](#) and described in [Table 29-43](#).

Return to the [Summary Table](#).

Inp Sel, LUT fns for Out7

**Figure 29-42. CLB\_OUTPUT\_LUT\_7 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									FN					IN2				IN1				IN0									
R/W1C-0h									R/W-0h					R/W-0h				R/W-0h				R/W-0h									

**Table 29-43. CLB\_OUTPUT\_LUT\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R/W1C	0h	Reserved
22-15	FN	R/W	0h	Output function for output LUT Reset type: SYSRSn
14-10	IN2	R/W	0h	Select value for IN2 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	IN1	R/W	0h	Select value for IN1 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	IN0	R/W	0h	Select value for IN0 of output LUT. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

**29.6.2.29 CLB\_HLC\_EVENT\_SEL Register (Offset = 3Ch) [reset = 0h]**

CLB\_HLC\_EVENT\_SEL is shown in [Figure 29-43](#) and described in [Table 29-44](#).

Return to the [Summary Table](#).

Event Selector register for the High Level controller

**Figure 29-43. CLB\_HLC\_EVENT\_SEL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED											EVENT3_SEL				
R-0-0h											R/W-0h				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVEN T3_SE L	EVENT2_SEL					EVENT1_SEL					EVENT0_SEL				
R/W- 0h	R/W-0h					R/W-0h					R/W-0h				

**Table 29-44. CLB\_HLC\_EVENT\_SEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R-0	0h	Reserved
19-15	EVENT3_SEL	R/W	0h	5 bit select value for EVENT3 of the High Level Controller. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
14-10	EVENT2_SEL	R/W	0h	5 bit select value for EVENT2 of the High Level Controller. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
9-5	EVENT1_SEL	R/W	0h	5 bit select value for EVENT1 of the High Level Controller. See the Static Switch Block Output Mux Table. Reset type: SYSRSn
4-0	EVENT0_SEL	R/W	0h	5 bit select value for EVENT0 of the High Level Controller. See the Static Switch Block Output Mux Table. Reset type: SYSRSn

### 29.6.3 CLB\_LOGIC\_CONTROL\_REGS Registers

Table 29-45 lists the CLB\_LOGIC\_CONTROL\_REGS registers. All register offset addresses not listed in Table 29-45 should be considered as reserved locations and the register contents should not be modified.

**Table 29-45. CLB\_LOGIC\_CONTROL\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	CLB_LOAD_EN	Global enable & indirect load enable control	EALLOW, LOCK (only Global Enable Bit is LOCK protected)	<a href="#">Go</a>
2h	CLB_LOAD_ADDR	Indirect address		<a href="#">Go</a>
4h	CLB_LOAD_DATA	Data for indirect loads		<a href="#">Go</a>
6h	CLB_INPUT_FILTER	Input filter selection for both edge detection and synchronizers	LOCK	<a href="#">Go</a>
8h	CLB_IN_MUX_SEL_0	Input selection to decide between Signals and GP register	LOCK	<a href="#">Go</a>
Ah	CLB_LCL_MUX_SEL_1	Input Mux selection for local mux	LOCK	<a href="#">Go</a>
Ch	CLB_LCL_MUX_SEL_2	Input Mux selection for local mux	LOCK	<a href="#">Go</a>
Eh	CLB_BUF_PTR	PUSH and PULL pointers		<a href="#">Go</a>
10h	CLB_GP_REG	General purpose register for CELL inputs		<a href="#">Go</a>
12h	CLB_OUT_EN	CELL output enable register		<a href="#">Go</a>
14h	CLB_GLBL_MUX_SEL_1	Global Mux select for CELL inputs	LOCK	<a href="#">Go</a>
16h	CLB_GLBL_MUX_SEL_2	Global Mux select for CELL inputs	LOCK	<a href="#">Go</a>
20h	CLB_INTR_TAG_REG	Interrupt Tag register		<a href="#">Go</a>
22h	CLB_LOCK	Lock control register	EALLOW	<a href="#">Go</a>
30h	CLB_DBG_R0	R0 of High level Controller		<a href="#">Go</a>
32h	CLB_DBG_R1	R1 of High level Controller		<a href="#">Go</a>
34h	CLB_DBG_R2	R2 of High level Controller		<a href="#">Go</a>
36h	CLB_DBG_R3	R3 of High level Controller		<a href="#">Go</a>
38h	CLB_DBG_C0	Count of Unit 0		<a href="#">Go</a>
3Ah	CLB_DBG_C1	Count of Unit 1		<a href="#">Go</a>
3Ch	CLB_DBG_C2	Count of Unit 2		<a href="#">Go</a>
3Eh	CLB_DBG_OUT	Outputs of various units in the Cell		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 29-46 shows the codes that are used for access types in this section.

**Table 29-46. CLB\_LOGIC\_CONTROL\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
R-0	R-0	Read Returns 0s
R-1	R-1	Read Returns 1s
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		

**Table 29-46. CLB\_LOGIC\_CONTROL\_REGS Access  
Type Codes (continued)**

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

**29.6.3.1 CLB\_LOAD\_EN Register (Offset = 0h) [reset = 0h]**

CLB\_LOAD\_EN is shown in [Figure 29-44](#) and described in [Table 29-47](#).

Return to the [Summary Table](#).

Global enable & indirect load enable control

**Figure 29-44. CLB\_LOAD\_EN Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED					STOP	GLOBAL_EN	LOAD_EN
R-0-0h					R/W-0h	R/W-0h	R/W-0h

**Table 29-47. CLB\_LOAD\_EN Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R-0	0h	Reserved
2	STOP	R/W	0h	This bit defines the behaviour of the sequential elements in the CELL during debug HALTs of the CPU. If this bit is set to 0, the debug HALT condition is ignored. Reset type: SYSRSn
1	GLOBAL_EN	R/W	0h	This bit is a global enable signal for the logic in the CELL. This also acts as a soft reset for the CELL logic. This bit is normally set after all the other configuration settings are completed. This bit is LOCK protected. Reset type: SYSRSn
0	LOAD_EN	R/W	0h	A write with this bit set to 1 will pulse the Load Enable signal for the indirect register loads in the CELL. Reset type: SYSRSn

### 29.6.3.2 CLB\_LOAD\_ADDR Register (Offset = 2h) [reset = 0h]

CLB\_LOAD\_ADDR is shown in [Figure 29-45](#) and described in [Table 29-48](#).

Return to the [Summary Table](#).

Indirect address

**Figure 29-45. CLB\_LOAD\_ADDR 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																										ADDR					
R-0-0h																										R/W-0h					

**Table 29-48. CLB\_LOAD\_ADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-0	ADDR	R/W	0h	These are the address bits used for writing to the indirect address space of the CELL. Reset type: SYSRSn

**29.6.3.3 CLB\_LOAD\_DATA Register (Offset = 4h) [reset = 0h]**

CLB\_LOAD\_DATA is shown in [Figure 29-46](#) and described in [Table 29-49](#).

Return to the [Summary Table](#).

Data for indirect loads

**Figure 29-46. CLB\_LOAD\_DATA 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
DATA																															
R/W-0h																															

**Table 29-49. CLB\_LOAD\_DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	0h	This register holds the 32-bit data for writing to the indirect address space of the CELL. Reset type: SYSRSn

**29.6.3.4 CLB\_INPUT\_FILTER Register (Offset = 6h) [reset = 0h]**

CLB\_INPUT\_FILTER is shown in [Figure 29-47](#) and described in [Table 29-50](#).

Return to the [Summary Table](#).

Input filter selection for both edge detection and synchronizers

**Figure 29-47. CLB\_INPUT\_FILTER Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
SYNC7	SYNC6	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0
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
FIN7		FIN6		FIN5		FIN4	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
FIN3		FIN2		FIN1		FIN0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 29-50. CLB\_INPUT\_FILTER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R-0	0h	Reserved
23	SYNC7	R/W	0h	Synchronizer Select Control for Input 7 Reset type: SYSRSn
22	SYNC6	R/W	0h	Synchronizer Select Control for Input 6 Reset type: SYSRSn
21	SYNC5	R/W	0h	Synchronizer Select Control for Input 5 Reset type: SYSRSn
20	SYNC4	R/W	0h	Synchronizer Select Control for Input 4 Reset type: SYSRSn
19	SYNC3	R/W	0h	Synchronizer Select Control for Input 3 Reset type: SYSRSn
18	SYNC2	R/W	0h	Synchronizer Select Control for Input 2 Reset type: SYSRSn
17	SYNC1	R/W	0h	Synchronizer Select Control for Input 1 Reset type: SYSRSn
16	SYNC0	R/W	0h	Synchronizer Select Control for Input 0 Reset type: SYSRSn
15-14	FIN7	R/W	0h	Input filter selection for CELL Input 7 2 bits are used to define the edge filtering . 00 : No filtering 01 : Rising edge detect 10 : Falling edge detect 11 : Any edge detect Reset type: SYSRSn



**Table 29-50. CLB\_INPUT\_FILTER Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	FIN6	R/W	0h	Input filter selection for CELL Input 6 2 bits are used to define the edge filtering . 00 : No filtering 01 : Rising edge detect 10 : Falling edge detect 11 : Any edge detect Reset type: SYSRSn
11-10	FIN5	R/W	0h	Input filter selection for CELL Input 5 2 bits are used to define the edge filtering . 00 : No filtering 01 : Rising edge detect 10 : Falling edge detect 11 : Any edge detect Reset type: SYSRSn
9-8	FIN4	R/W	0h	Input filter selection for CELL Input 4 2 bits are used to define the edge filtering . 00 : No filtering 01 : Rising edge detect 10 : Falling edge detect 11 : Any edge detect Reset type: SYSRSn
7-6	FIN3	R/W	0h	Input filter selection for CELL Input 3 2 bits are used to define the edge filtering . 00 : No filtering 01 : Rising edge detect 10 : Falling edge detect 11 : Any edge detect Reset type: SYSRSn
5-4	FIN2	R/W	0h	Input filter selection for CELL Input 2 2 bits are used to define the edge filtering . 00 : No filtering 01 : Rising edge detect 10 : Falling edge detect 11 : Any edge detect Reset type: SYSRSn
3-2	FIN1	R/W	0h	Input filter selection for CELL Input 1 2 bits are used to define the edge filtering . 00 : No filtering 01 : Rising edge detect 10 : Falling edge detect 11 : Any edge detect Reset type: SYSRSn
1-0	FIN0	R/W	0h	Input filter selection for CELL Input 0 2 bits are used to define the edge filtering . 00 : No filtering 01 : Rising edge detect 10 : Falling edge detect 11 : Any edge detect Reset type: SYSRSn

**29.6.3.5 CLB\_IN\_MUX\_SEL\_0 Register (Offset = 8h) [reset = 0h]**

CLB\_IN\_MUX\_SEL\_0 is shown in [Figure 29-48](#) and described in [Table 29-51](#).

Return to the [Summary Table](#).

Input selection to decide between Signals and GP register

**Figure 29-48. CLB\_IN\_MUX\_SEL\_0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
SEL_GP_IN_7	SEL_GP_IN_6	SEL_GP_IN_5	SEL_GP_IN_4	SEL_GP_IN_3	SEL_GP_IN_2	SEL_GP_IN_1	SEL_GP_IN_0
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 29-51. CLB\_IN\_MUX\_SEL\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	SEL_GP_IN_7	R/W	0h	Select control for Input 7 to decide between external input and CLB_GP_REG[7] 0 : Input comes from selected external input 1 : Input comes from CLB_GP_REG[7] Reset type: SYSRSn
6	SEL_GP_IN_6	R/W	0h	Select control for Input 6 to decide between external input and CLB_GP_REG[6] 0 : Input comes from selected external input 1 : Input comes from CLB_GP_REG[6] Reset type: SYSRSn
5	SEL_GP_IN_5	R/W	0h	Select control for Input 5 to decide between external input and CLB_GP_REG[5] 0 : Input comes from selected external input 1 : Input comes from CLB_GP_REG[5] Reset type: SYSRSn
4	SEL_GP_IN_4	R/W	0h	Select control for Input 4 to decide between external input and CLB_GP_REG[4] 0 : Input comes from selected external input 1 : Input comes from CLB_GP_REG[4] Reset type: SYSRSn
3	SEL_GP_IN_3	R/W	0h	Select control for Input 3 to decide between external input and CLB_GP_REG[3] 0 : Input comes from selected external input 1 : Input comes from CLB_GP_REG[3] Reset type: SYSRSn

**Table 29-51. CLB\_IN\_MUX\_SEL\_0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	SEL_GP_IN_2	R/W	0h	Select control for Input 2 to decide between external input and CLB_GP_REG[2] 0 : Input comes from selected external input 1 : Input comes from CLB_GP_REG[2] Reset type: SYSRSn
1	SEL_GP_IN_1	R/W	0h	Select control for Input 1 to decide between external input and CLB_GP_REG[1] 0 : Input comes from selected external input 1 : Input comes from CLB_GP_REG[1] Reset type: SYSRSn
0	SEL_GP_IN_0	R/W	0h	Select control for Input 0 to decide between external input and CLB_GP_REG[0] 0 : Input comes from selected external input 1 : Input comes from CLB_GP_REG[0] Reset type: SYSRSn

**29.6.3.6 CLB\_LCL\_MUX\_SEL\_1 Register (Offset = Ah) [reset = 0h]**

CLB\_LCL\_MUX\_SEL\_1 is shown in [Figure 29-49](#) and described in [Table 29-52](#).

Return to the [Summary Table](#).

Input Mux selection for local mux

**Figure 29-49. CLB\_LCL\_MUX\_SEL\_1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED				LCL_MUX_SEL_IN_3			
R-0-0h				R/W-0h			
15	14	13	12	11	10	9	8
LCL_MUX_SEL_IN_3	LCL_MUX_SEL_IN_2					LCL_MUX_SEL_IN_1	
R/W-0h		R/W-0h			R/W-0h		
7	6	5	4	3	2	1	0
LCL_MUX_SEL_IN_1			LCL_MUX_SEL_IN_0				
R/W-0h			R/W-0h				

**Table 29-52. CLB\_LCL\_MUX\_SEL\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R-0	0h	Reserved
19-15	LCL_MUX_SEL_IN_3	R/W	0h	5 bit MUX Select for Local MUX control for Input 3 See Local Signals and Mux Selection Table Reset type: SYSRSn
14-10	LCL_MUX_SEL_IN_2	R/W	0h	5 bit MUX Select for Local MUX control for Input 2 See Local Signals and Mux Selection Table Reset type: SYSRSn
9-5	LCL_MUX_SEL_IN_1	R/W	0h	5 bit MUX Select for Local MUX control for Input 1 See Local Signals and Mux Selection Table Reset type: SYSRSn
4-0	LCL_MUX_SEL_IN_0	R/W	0h	5 bit MUX Select for Local MUX control for Input 0 See Local Signals and Mux Selection Table Reset type: SYSRSn

**29.6.3.7 CLB\_LCL\_MUX\_SEL\_2 Register (Offset = Ch) [reset = 0h]**

 CLB\_LCL\_MUX\_SEL\_2 is shown in [Figure 29-50](#) and described in [Table 29-53](#).

 Return to the [Summary Table](#).

Input Mux selection for local mux

**Figure 29-50. CLB\_LCL\_MUX\_SEL\_2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED				LCL_MUX_SEL_IN_7			
R-0-0h				R/W-0h			
15	14	13	12	11	10	9	8
LCL_MUX_SEL_IN_7	LCL_MUX_SEL_IN_6					LCL_MUX_SEL_IN_5	
R/W-0h		R/W-0h			R/W-0h		
7	6	5	4	3	2	1	0
LCL_MUX_SEL_IN_5			LCL_MUX_SEL_IN_4				
R/W-0h			R/W-0h				

**Table 29-53. CLB\_LCL\_MUX\_SEL\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R-0	0h	Reserved
19-15	LCL_MUX_SEL_IN_7	R/W	0h	5 bit MUX Select for Local MUX control for Input 7 See Local Signals and Mux Selection Table Reset type: SYSRSn
14-10	LCL_MUX_SEL_IN_6	R/W	0h	5 bit MUX Select for Local MUX control for Input 6 See Local Signals and Mux Selection Table Reset type: SYSRSn
9-5	LCL_MUX_SEL_IN_5	R/W	0h	5 bit MUX Select for Local MUX control for Input 5 See Local Signals and Mux Selection Table Reset type: SYSRSn
4-0	LCL_MUX_SEL_IN_4	R/W	0h	5 bit MUX Select for Local MUX control for Input 4 See Local Signals and Mux Selection Table Reset type: SYSRSn

### 29.6.3.8 CLB\_BUF\_PTR Register (Offset = Eh) [reset = 0h]

CLB\_BUF\_PTR is shown in [Figure 29-51](#) and described in [Table 29-54](#).

Return to the [Summary Table](#).

PUSH and PULL pointers

**Figure 29-51. CLB\_BUF\_PTR 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								PUSH								RESERVED								PULL							
R-0-0h								R/W-0h								R-0-0h								R/W-0h							

**Table 29-54. CLB\_BUF\_PTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R-0	0h	Reserved
23-16	PUSH	R/W	0h	8 bit pointer which indicates the number of data values which have been pulled from the buffer by the High Level Controller. This counter will wrap around after 0xff. The Least significant 2 bits are used as the actual pointer for the operation. Reset type: SYSRSn
15-8	RESERVED	R-0	0h	Reserved
7-0	PULL	R/W	0h	8 bit pointer which indicates the number of data values that have been written by the High Level controller into the buffer. The Least significant 2 bits are used as the actual pointer for the operation. Reset type: SYSRSn

**29.6.3.9 CLB\_GP\_REG Register (Offset = 10h) [reset = 0h]**

CLB\_GP\_REG is shown in [Figure 29-52](#) and described in [Table 29-55](#).

Return to the [Summary Table](#).

General purpose register for CELL inputs

**Figure 29-52. CLB\_GP\_REG 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																REG															
R-0-0h																R/W-0h															

**Table 29-55. CLB\_GP\_REG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7-0	REG	R/W	0h	8 bits which are directly connected to the 8 inputs of the CELL if that corresponding bit is selected in the CLB_IN_MUX_SEL_0 register Reset type: SYSRSn

**29.6.3.10 CLB\_OUT\_EN Register (Offset = 12h) [reset = 0h]**

CLB\_OUT\_EN is shown in [Figure 29-53](#) and described in [Table 29-56](#).

Return to the [Summary Table](#).

CELL output enable register

**Figure 29-53. CLB\_OUT\_EN 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
OUT0																															
R/W-0h																															

**Table 29-56. CLB\_OUT\_EN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	OUT0	R/W	0h	32 bits which are directly driven out as OUTPUT_EN signals. These are meant to be used by logic outside the CLB. Reset type: SYSRSn



### 29.6.3.11 CLB\_GLBL\_MUX\_SEL\_1 Register (Offset = 14h) [reset = 0h]

CLB\_GLBL\_MUX\_SEL\_1 is shown in [Figure 29-54](#) and described in [Table 29-57](#).

Return to the [Summary Table](#).

Global Mux select for CELL inputs

**Figure 29-54. CLB\_GLBL\_MUX\_SEL\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				GLBL_MUX_SEL_IN_3							GLBL_MUX_SEL_IN_2				
R-0-0h				R/W-0h							R/W-0h				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GLBL_MUX_SEL_IN_2		GLBL_MUX_SEL_IN_1							GLBL_MUX_SEL_IN_0						
R/W-0h		R/W-0h							R/W-0h						

**Table 29-57. CLB\_GLBL\_MUX\_SEL\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R-0	0h	Reserved
27-21	GLBL_MUX_SEL_IN_3	R/W	0h	7 bit MUX Select for Global MUX control for Input 3 See Global Signals and Mux Selection Table Reset type: SYSRSn
20-14	GLBL_MUX_SEL_IN_2	R/W	0h	7 bit MUX Select for Global MUX control for Input 2 See Global Signals and Mux Selection Table Reset type: SYSRSn
13-7	GLBL_MUX_SEL_IN_1	R/W	0h	7 bit MUX Select for Global MUX control for Input 1 See Global Signals and Mux Selection Table Reset type: SYSRSn
6-0	GLBL_MUX_SEL_IN_0	R/W	0h	7 bit MUX Select for Global MUX control for Input 0 See Global Signals and Mux Selection Table Reset type: SYSRSn

**29.6.3.12 CLB\_GLBL\_MUX\_SEL\_2 Register (Offset = 16h) [reset = 0h]**

CLB\_GLBL\_MUX\_SEL\_2 is shown in [Figure 29-55](#) and described in [Table 29-58](#).

Return to the [Summary Table](#).

Global Mux select for CELL inputs

**Figure 29-55. CLB\_GLBL\_MUX\_SEL\_2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				GLBL_MUX_SEL_IN_7							GLBL_MUX_SEL_IN_6				
R-0-0h				R/W-0h							R/W-0h				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GLBL_MUX_S EL_IN_6		GLBL_MUX_SEL_IN_5							GLBL_MUX_SEL_IN_4						
R/W-0h		R/W-0h							R/W-0h						

**Table 29-58. CLB\_GLBL\_MUX\_SEL\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R-0	0h	Reserved
27-21	GLBL_MUX_SEL_IN_7	R/W	0h	7 bit MUX Select for Global MUX control for Input 7 See Global Signals and Mux Selection Table Reset type: SYSRSn
20-14	GLBL_MUX_SEL_IN_6	R/W	0h	7 bit MUX Select for Global MUX control for Input 6 See Global Signals and Mux Selection Table Reset type: SYSRSn
13-7	GLBL_MUX_SEL_IN_5	R/W	0h	7 bit MUX Select for Global MUX control for Input 5 See Global Signals and Mux Selection Table Reset type: SYSRSn
6-0	GLBL_MUX_SEL_IN_4	R/W	0h	7 bit MUX Select for Global MUX control for Input 4 See Global Signals and Mux Selection Table Reset type: SYSRSn

**29.6.3.13 CLB\_INTR\_TAG\_REG Register (Offset = 20h) [reset = 0h]**

CLB\_INTR\_TAG\_REG is shown in [Figure 29-56](#) and described in [Table 29-59](#).

Return to the [Summary Table](#).

Interrupt Tag register

**Figure 29-56. CLB\_INTR\_TAG\_REG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				TAG			
R-0-0h				R/W-0h			

**Table 29-59. CLB\_INTR\_TAG\_REG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-6	RESERVED	R-0	0h	Reserved
5-0	TAG	R/W	0h	6 bits which are used by the High Level Controller to set a tag value on flagging interrupts. . This can be cleared through the VBUS interface since it is writeable through the VBUS. Reset type: SYSRSn

**29.6.3.14 CLB\_LOCK Register (Offset = 22h) [reset = 0h]**

CLB\_LOCK is shown in [Figure 29-57](#) and described in [Table 29-60](#).

Return to the [Summary Table](#).

Lock control register

**Figure 29-57. CLB\_LOCK Register**

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

**Table 29-60. CLB\_LOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R/W	0h	These 16 bits act as a key to enable writes to Bit 0 of this register. The only time a '1' can be written to Bit 0 is by a single 32-bit write where bits 31:16 equal 0x5a5a and bit 0 is '1'. All other writes are ignored including separate 16-bit writes. This is EALLOW protected. Reset type: SYSRSn
15-1	RESERVED	R-0	0h	Reserved
0	LOCK	R/W	0h	This bit is used as a one-time write bit (Set Once). Once it is set to '1', only a reset (SYSRSN 0) will clear this bit back to 0. Reset type: SYSRSn

**29.6.3.15 CLB\_DBG\_R0 Register (Offset = 30h) [reset = 0h]**

CLB\_DBG\_R0 is shown in [Figure 29-58](#) and described in [Table 29-61](#).

Return to the [Summary Table](#).

R0 of High level Controller

**Figure 29-58. CLB\_DBG\_R0 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
																	DBG														
																	R-0h														

**Table 29-61. CLB\_DBG\_R0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DBG	R	0h	CLB_DBG_R0 Reset type: SYSRSn

**29.6.3.16 CLB\_DBG\_R1 Register (Offset = 32h) [reset = 0h]**

CLB\_DBG\_R1 is shown in [Figure 29-59](#) and described in [Table 29-62](#).

Return to the [Summary Table](#).

R1 of High level Controller

**Figure 29-59. CLB\_DBG\_R1 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
																	DBG														
																	R-0h														

**Table 29-62. CLB\_DBG\_R1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DBG	R	0h	CLB_DBG_R1 Reset type: SYSRSn

**29.6.3.17 CLB\_DBG\_R2 Register (Offset = 34h) [reset = 0h]**

CLB\_DBG\_R2 is shown in [Figure 29-60](#) and described in [Table 29-63](#).

Return to the [Summary Table](#).

R2 of High level Controller

**Figure 29-60. CLB\_DBG\_R2 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
																	DBG														
																	R-0h														

**Table 29-63. CLB\_DBG\_R2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DBG	R	0h	CLB_DBG_R2 Reset type: SYSRSn

**29.6.3.18 CLB\_DBG\_R3 Register (Offset = 36h) [reset = 0h]**

CLB\_DBG\_R3 is shown in [Figure 29-61](#) and described in [Table 29-64](#).

Return to the [Summary Table](#).

R3 of High level Controller

**Figure 29-61. CLB\_DBG\_R3 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
																	DBG														
																	R-0h														

**Table 29-64. CLB\_DBG\_R3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DBG	R	0h	CLB_DBG_R3 Reset type: SYSRSn



**29.6.3.19 CLB\_DBG\_C0 Register (Offset = 38h) [reset = 0h]**

CLB\_DBG\_C0 is shown in [Figure 29-62](#) and described in [Table 29-65](#).

Return to the [Summary Table](#).

Count of Unit 0

**Figure 29-62. CLB\_DBG\_C0 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
																	DBG														
																	R-0h														

**Table 29-65. CLB\_DBG\_C0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DBG	R	0h	CLB_DBG_C0 Reset type: SYSRSn

**29.6.3.20 CLB\_DBG\_C1 Register (Offset = 3Ah) [reset = 0h]**

CLB\_DBG\_C1 is shown in [Figure 29-63](#) and described in [Table 29-66](#).

Return to the [Summary Table](#).

Count of Unit 1

**Figure 29-63. CLB\_DBG\_C1 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
																	DBG														
R-0h																															

**Table 29-66. CLB\_DBG\_C1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DBG	R	0h	CLB_DBG_C1 Reset type: SYSRSn

**29.6.3.21 CLB\_DBG\_C2 Register (Offset = 3Ch) [reset = 0h]**

CLB\_DBG\_C2 is shown in [Figure 29-64](#) and described in [Table 29-67](#).

Return to the [Summary Table](#).

Count of Unit 2

**Figure 29-64. CLB\_DBG\_C2 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
																	DBG														
																	R-0h														

**Table 29-67. CLB\_DBG\_C2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DBG	R	0h	CLB_DBG_C2 Reset type: SYSRSn

**29.6.3.22 CLB\_DBG\_OUT Register (Offset = 3Eh) [reset = 00010100h]**

CLB\_DBG\_OUT is shown in [Figure 29-65](#) and described in [Table 29-68](#).

Return to the [Summary Table](#).

Outputs of various units in the Cell

**Figure 29-65. CLB\_DBG\_OUT Register**

31		30		29		28		27		26		25		24	
OUT7		OUT6		OUT5		OUT4		OUT3		OUT2		OUT1		OUT0	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h	
23		22		21		20		19		18		17		16	
LUT42_OUT		FSM2_LUTOU T		FSM2_S1		FSM2_S0		COUNT2_MAT CH1		COUNT2_ZER O		COUNT2_MAT CH2		RESERVED	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-1-1h	
15		14		13		12		11		10		9		8	
LUT41_OUT		FSM1_LUTOU T		FSM1_S1		FSM1_S0		COUNT1_MAT CH1		COUNT1_ZER O		COUNT1_MAT CH2		RESERVED	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-1-1h	
7		6		5		4		3		2		1		0	
LUT40_OUT		FSM0_LUTOU T		FSM0_S1		FSM0_S0		COUNT0_MAT CH1		COUNT0_ZER O		COUNT0_MAT CH2		RESERVED	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0-0h	

**Table 29-68. CLB\_DBG\_OUT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	OUT7	R	0h	CELL Output 7 Reset type: SYSRSn
30	OUT6	R	0h	CELL Output 6 Reset type: SYSRSn
29	OUT5	R	0h	CELL Output 5 Reset type: SYSRSn
28	OUT4	R	0h	CELL Output 4 Reset type: SYSRSn
27	OUT3	R	0h	CELL Output 3 Reset type: SYSRSn
26	OUT2	R	0h	CELL Output 2 Reset type: SYSRSn
25	OUT1	R	0h	CELL Output 1 Reset type: SYSRSn
24	OUT0	R	0h	CELL Output 0 Reset type: SYSRSn
23	LUT42_OUT	R	0h	LUT4_OUT UNIT 2 Reset type: SYSRSn
22	FSM2_LUTOU	R	0h	FSM_LUT_OUT UNIT 2 Reset type: SYSRSn
21	FSM2_S1	R	0h	FSM_S1 UNIT 2 Reset type: SYSRSn
20	FSM2_S0	R	0h	FSM_S0 UNIT 2 Reset type: SYSRSn
19	COUNT2_MATCH1	R	0h	COUNT_MATCH1 UNIT 2 Reset type: SYSRSn

**Table 29-68. CLB\_DBG\_OUT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	COUNT2_ZERO	R	0h	COUNT_ZERO UNIT 2 Reset type: SYSRSn
17	COUNT2_MATCH2	R	0h	COUNT_MATCH2 UNIT 2 Reset type: SYSRSn
16	RESERVED	R-1	1h	Reserved
15	LUT41_OUT	R	0h	LUT4_OUT UNIT 1 Reset type: SYSRSn
14	FSM1_LUTOUT	R	0h	FSM_LUT_OUT UNIT 1 Reset type: SYSRSn
13	FSM1_S1	R	0h	FSM_S1 UNIT 1 Reset type: SYSRSn
12	FSM1_S0	R	0h	FSM_S0 UNIT 1 Reset type: SYSRSn
11	COUNT1_MATCH1	R	0h	COUNT_MATCH1 UNIT 1 Reset type: SYSRSn
10	COUNT1_ZERO	R	0h	COUNT_ZERO UNIT 1 Reset type: SYSRSn
9	COUNT1_MATCH2	R	0h	COUNT_MATCH2 UNIT 1 Reset type: SYSRSn
8	RESERVED	R-1	1h	Reserved
7	LUT40_OUT	R	0h	LUT4_OUT UNIT 0 Reset type: SYSRSn
6	FSM0_LUTOUT	R	0h	FSM_LUT_OUT UNIT 0 Reset type: SYSRSn
5	FSM0_S1	R	0h	FSM_S1 UNIT 0 Reset type: SYSRSn
4	FSM0_S0	R	0h	FSM_S0 UNIT 0 Reset type: SYSRSn
3	COUNT0_MATCH1	R	0h	COUNT_MATCH1 UNIT 0 Reset type: SYSRSn
2	COUNT0_ZERO	R	0h	COUNT_ZERO UNIT 0 Reset type: SYSRSn
1	COUNT0_MATCH2	R	0h	COUNT_MATCH2 UNIT 0 Reset type: SYSRSn
0	RESERVED	R-0	0h	Reserved

### 29.6.4 CLB\_DATA\_EXCHANGE\_REGS Registers

Table 29-69 lists the CLB\_DATA\_EXCHANGE\_REGS registers. All register offset addresses not listed in Table 29-69 should be considered as reserved locations and the register contents should not be modified.

**Table 29-69. CLB\_DATA\_EXCHANGE\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h + formula	CLB_PUSH_y	CLB_PUSH FIFO Registers (from HLC)		<a href="#">Go</a>
100h + formula	CLB_PULL_y	CLB_PULL FIFO Registers (TO HLC)		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 29-70 shows the codes that are used for access types in this section.

**Table 29-70. CLB\_DATA\_EXCHANGE\_REGS Access Type Codes**

Access Type	Code	Description
<b>Read Type</b>		
R	R	Read
<b>Write Type</b>		
W	W	Write
<b>Reset or Default Value</b>		
-n		Value after reset or the default value
<b>Register Array Variables</b>		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 29.6.4.1 CLB\_PUSH\_y Register (Offset = 0h + formula) [reset = 0h]

CLB\_PUSH\_y is shown in [Figure 29-66](#) and described in [Table 29-71](#).

Return to the [Summary Table](#).

CLB\_PUSH FIFO Registers (from HLC)

Offset = 0h + (y \* 2h); where y = 0h to 3h

**Figure 29-66. CLB\_PUSH\_y 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
																PUSH															
																R-0h															

**Table 29-71. CLB\_PUSH\_y Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	PUSH	R	0h	FIFO TO System From CLB Reset type: SYSRSn

**29.6.4.2 CLB\_PULL\_y Register (Offset = 100h + formula) [reset = 0h]**

CLB\_PULL\_y is shown in [Figure 29-67](#) and described in [Table 29-72](#).

Return to the [Summary Table](#).

CLB\_PULL FIFO Registers (TO HLC)

Offset = 100h + (y \* 2h); where y = 0h to 3h

**Figure 29-67. CLB\_PULL\_y 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						
																	PULL																				
R/W-0h																																					

**Table 29-72. CLB\_PULL\_y Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	PULL	R/W	0h	FIFO From system TO CLB Reset type: SYSRSn



## 29.6.5 Register to Driverlib Function Mapping

**Table 29-73. CLB Registers to Driverlib Functions**

File	Driverlib Function
<b>COUNT_RESET</b>	
clb.h	CLB_selectCounterInputs
<b>COUNT_MODE_1</b>	
clb.h	CLB_selectCounterInputs
<b>COUNT_MODE_0</b>	
clb.h	CLB_selectCounterInputs
<b>COUNT_EVENT</b>	
clb.h	CLB_selectCounterInputs
<b>FSM_EXTRA_IN0</b>	
clb.h	CLB_selectFSMInputs
<b>FSM_EXTERNAL_IN0</b>	
clb.h	CLB_selectFSMInputs
<b>FSM_EXTERNAL_IN1</b>	
clb.h	CLB_selectFSMInputs
<b>FSM_EXTRA_IN1</b>	
clb.h	CLB_selectFSMInputs
<b>LUT4_IN0</b>	
clb.h	CLB_selectLUT4Inputs
<b>LUT4_IN1</b>	
clb.h	CLB_selectLUT4Inputs
<b>LUT4_IN2</b>	
clb.h	CLB_selectLUT4Inputs
<b>LUT4_IN3</b>	
clb.h	CLB_selectLUT4Inputs
<b>FSM_LUT_FN1_0</b>	
clb.h	CLB_configFSMLUTFunction
<b>FSM_LUT_FN2</b>	
clb.h	CLB_configFSMLUTFunction
<b>LUT4_FN1_0</b>	
clb.h	CLB_configLUT4Function
<b>LUT4_FN2</b>	
clb.h	CLB_configLUT4Function
<b>FSM_NEXT_STATE_0</b>	
clb.h	CLB_configFSMNextState
<b>FSM_NEXT_STATE_1</b>	
clb.h	CLB_configFSMNextState
<b>FSM_NEXT_STATE_2</b>	
clb.h	CLB_configFSMNextState
<b>MISC_CONTROL</b>	
clb.h	CLB_configMiscCtrlModes
<b>OUTPUT_LUT_0</b>	
clb.h	CLB_configOutputLUT
<b>OUTPUT_LUT_1</b>	
-	See OUTPUT_LUT_0
<b>OUTPUT_LUT_2</b>	
-	See OUTPUT_LUT_0

**Table 29-73. CLB Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>OUTPUT_LUT_3</b>	
-	See OUTPUT_LUT_0
<b>OUTPUT_LUT_4</b>	
-	See OUTPUT_LUT_0
<b>OUTPUT_LUT_5</b>	
-	See OUTPUT_LUT_0
<b>OUTPUT_LUT_6</b>	
-	See OUTPUT_LUT_0
<b>OUTPUT_LUT_7</b>	
-	See OUTPUT_LUT_0
<b>HLC_EVENT_SEL</b>	
clb.h	CLB_configHLCEventSelect
<b>LOAD_EN</b>	
clb.h	CLB_enableCLB
clb.h	CLB_writeInterface
<b>LOAD_ADDR</b>	
clb.h	CLB_writeInterface
<b>LOAD_DATA</b>	
clb.h	CLB_writeInterface
<b>INPUT_FILTER</b>	
clb.h	CLB_selectInputFilter
clb.h	CLB_enableSynchronization
clb.h	CLB_disableSynchronization
<b>IN_MUX_SEL_0</b>	
clb.h	CLB_configGPInputMux
<b>LCL_MUX_SEL_1</b>	
clb.h	CLB_configLocalInputMux
<b>LCL_MUX_SEL_2</b>	
clb.h	CLB_configLocalInputMux
<b>BUF_PTR</b>	
clb.c	CLB_clearFIFOs
<b>GP_REG</b>	
clb.h	CLB_setGPREG
<b>OUT_EN</b>	
clb.h	CLB_setOutputMask
<b>GLBL_MUX_SEL_1</b>	
clb.h	CLB_configGlobalInputMux
<b>GLBL_MUX_SEL_2</b>	
clb.h	CLB_configGlobalInputMux
<b>INTR_TAG_REG</b>	
clb.h	CLB_getInterruptTag
clb.h	CLB_clearInterruptTag
<b>LOCK</b>	
clb.h	CLB_enableLock
<b>DBG_OUT</b>	
clb.h	CLB_getOutputStatus

**Table 29-73. CLB Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>PUSH</b>	
clb.c	CLB_readFIFOs
<b>PULL</b>	
clb.c	CLB_clearFIFOs
clb.c	CLB_writeFIFOs

## Revision History

<b>Changes from January 17, 2019 to August 10, 2019</b>	<b>Page</b>
• <a href="#">Section 3.3</a> : Revised this section with an itemized list. ....	89
• <a href="#">Table 4-2</a> : Revised text in Step 4.....	609
• <a href="#">Table 4-4</a> : Removed rRw 8 and the accompanying reference. ....	610
• <a href="#">Table 4-14</a> : Removed the last row referencing the PLC. ....	618
• <a href="#">Table 4-16</a> : Removed rows containing the text "PLC.".....	619
• <a href="#">Table 4-17</a> : Removed rows containing "PLC." .....	619
• <a href="#">Table 4-40</a> : Deleted row containing "PLC boot has started.".....	639
• <a href="#">Figure 10-3</a> : Revised the diagram. ....	1367
• <a href="#">Section 13.4.2</a> : Revised the text in this section for better clarity. ....	1479
• <a href="#">Section 15.2.3</a> : Added the EPWMSYNCPER Signal section. ....	1677
• <a href="#">Section 16.7.1</a> EPWMSYNCO updated to EPWMSYNCPER. ....	1695
• <a href="#">Section 16.7.2</a> : Added the Synchronizer, Digital Filter, and Latch Delays section. ....	1695
• <a href="#">Section 16.7.3</a> : Added the Calibrating the CMPSS section. ....	1696
• <a href="#">Section 18.1</a> : Added a Note to the end of the Introduction.....	1811
• <a href="#">Section 18.1.1</a> : Added EPWMSYNCPER description to the PWMxSYNCl and EPWMxSYNCO bulleted llist item. ....	1813
• <a href="#">Table 18-2</a> : Added EPWMxSYNCPER .....	1821
• <a href="#">Section 18.4.3.3</a> : Added snippet about SYNCPER.....	1824
• <a href="#">Figure 19-8</a> : Revised the graphic. ....	1825
• <a href="#">Figure 18-54</a> : Revised the Event Filtering figure.....	1877
• <a href="#">Section 21.8</a> : Modified the text in the first paragraph. ....	2160
• <a href="#">Section 27.1</a> : Removed K-Line comment.....	2473
• <a href="#">Chapter 29</a> : Added the Configurable Logic Block chapter.....	2684

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale ([www.ti.com/legal/termsofsale.html](http://www.ti.com/legal/termsofsale.html)) or other applicable terms available either on [ti.com](http://ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2019, Texas Instruments Incorporated